ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA250812

A Conditional Generative Diffusion Model for Spatio-Temporal Data

Giulio Loddi[a,*], **Alessandro Betti**[a] **and Fabio Pinelli**[a,b]

[a]IMT School for Advanced Studies Lucca, Lucca, Italy
[b]ISTI-CNR, Pisa, Italy

Abstract. Diffusion models have become widely used for generating text, image, video, and audio. In recent years, these models have also been introduced into the time series domain for tasks such as forecasting, imputation, and generation. An interesting application is conditional generation with respect to metadata, enabling the synthesis of data sequences that match specified conditions. In the present work, we focus on spatio-temporal data and more precisely on time series data that also exhibit a spatial nature—for instance, measurements from sensor networks, traffic flows, mobile network usage across different areas. However, current approaches for time series conditional generation often neglect spatial autocorrelation. In this work, we extend the well-known DIFFWAVE model to address this challenge by directly taking into account the spatial nature of the data in the denoising process of the diffusion model. We evaluate our approach on a large and complex real-world dataset from the NETMOB 2023 data challenge, which collects mobile network usage of different mobile applications across urban areas. Our results demonstrate that, in addition to achieving competitive performance across all evaluated metrics, our approach is also able to correctly capture the spatial autocorrelation present in the real data.

1 Introduction

Spatio-temporal data such as mobile traffic, urban trajectories, and climate records are pervasive in modern life. Access to such data is crucial for understanding complex phenomena, training machine learning models and taking data-informed decisions. However, real data are often costly to obtain, privacy-sensitive, or controlled by a limited number of organizations. Synthetic data generation offers a promising alternative, enabling broader access to realistic data while preserving sensitive information and reducing collection costs.

Diffusion models have recently emerged as powerful generative tools for images, audio, video, and time series. In the time series domain, diffusion models have been applied primarily to forecasting and imputation tasks. However, their use for pure generation, particularly when conditioned on external metadata, remains limited [30].

Even less attention has been given to the generation of spatio-temporal sequences—that is, time series that are distributed over space, such as sensor readings across geographic regions or network traffic across urban areas. In such settings, generating realistic temporal patterns is not enough; preserving spatial relationships between locations is equally crucial. Ignoring spatial dependencies can produce sequences that are temporally coherent but spatially implausible. For instance, we reported in Figure 1 the Instagram mobile traffic volume recorded every 15 minutes in a selected location and eight spatially close locations in Paris. We can notice a significant correlation between the time series, an aspect that current methods often fail to capture, yet is crucial for generating spatially realistic data.

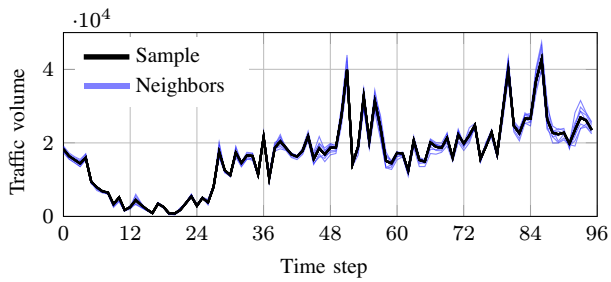

Figure 1: Daily time series of Instagram traffic volume recorded every 15 minutes for a selected urban area and its eight neighboring in Paris.

To address this, we propose DIFFWAVE3D1D, a spatio-temporal extension of the popular diffusion-based audio synthesis model DIFFWAVE [13]. Our architecture combines temporal and spatial modeling via a hybrid 3D+1D convolutional design, capturing local temporal dynamics and spatial correlations without relying on more complex and computationally demanding attention-like mechanisms.

We test our approach on a challenging and rich real-world dataset—the NETMOB23 dataset—which provides high-resolution mobile network usage data across urban areas. We show that, for grid-structured data, a convolutional approach is not only computationally efficient but also competitive with attention-based methods like CSDI [27].

To summarize, our main contributions are:

- We introduce DIFFWAVE3D1D, a conditional generative diffusion model tailored to grid-based and high-resolution spatio-temporal data;
- We demonstrate that a purely convolutional design is significantly more efficient than spatio-temporal attention while matching or exceeding the realism, spatial fidelity, and Train on Synthetic Test on Real (TSTR) accuracy of the attention-based state-of-the-art CSDI
- We conduct for the first time a comprehensive evaluation of generative models on the NETMOB23 dataset, benchmarking against strong baselines using metrics that assess realism, spatial coherence, and downstream utility.

* Corresponding Author. Email: giulio.loddi@imtlucca.it.

The rest of the paper is structured as follows. In Section 2 we summarize recent generative approaches for spatio-temporal data. Section 3 revisits the fundamentals of diffusion-based generative modelling, giving the loss function typically optimised by denoising probabilistic diffusion models. Section 4 describes the DIFFWAVE denoiser architecture, the common backbone of all models tested in this work. In Section 5 we introduce our variations to the DIFFWAVE architecture to handle correct modeling of spatial autocorrelation. In Section 6 we define the metrics used to evalutate our method and compare it both visually and quantitatively to the chosen baselines.

2 Related Works

Generative methods have witnessed remarkable advancements in recent years, enabling machines to generate content that closely resembles human creations. In particular, Generative Adversarial Networks (GANS) [8] have significantly contributed to various generative domains, including image, video, and time series data [11, 32, 16]. However, GANs are known to suffer from training instability and mode collapse [14, 28], which has led to the development of diffusion models as more stable and reliable alternatives [6, 9, 10].

In the time series domain, diffusion models are now considered the state-of-the-art approach. They have proven to be effective for classical tasks such as forecasting [5, 25, 24] and imputation [27, 1]. However, their application to purely generative tasks is still limited. For instance, SSSD-ECG [2], built on the Structured State Space Diffusion (SSSD) architecture [1], generates synthetic electrocardiograms conditioned on binary statements. Time Weaver [22] is another conditional generative diffusion model, extending the imputation frameworks CSDI [27] and SSSD [1]. A notable contribution of this work is its flexible metadata embedding module, which enables the encoding of rich and heterogeneous conditions (e.g., sensor type, weather condition, day of collection) during the generation process.

In the context of spatio-temporal data analysis, incorporating spatial information has led to improvements in various tasks, including forecasting, imputation, and anomaly detection [12]. Recent methods often employ spatial attention mechanisms and message-passing modules to capture complex spatio-temporal dependencies [4, 20, 18].

To the best of our knowledge, no purely generative conditional diffusion model has yet been designed to explicitly model spatial relationships and produce spatially coherent temporal data.

We also note that, unlike most commonly used benchmark datasets in the spatio-temporal domain, such as METR-LA [17] (207 nodes), PEMS-BAY [17] (325 nodes), or AQI-36 [31] (36 nodes), which have irregular topologies and contain only hundreds of spatial nodes, the NETMOB23 dataset employed in this work features a dense, grid-based spatial structure, containing over 68,000 spatial locations. This represents a two-order-of-magnitude increase in spatial size compared to standard benchmarks. We argue that working on such a large-scale, high-resolution dataset not only poses new challenges for generative modeling but also constitutes a significant contribution to the advancement of spatio-temporal generative methods.

3 Background on Diffusion Models

Diffusion models synthesize data by reversing a gradual noising process (denoising). Consider a *clean* sample $\boldsymbol{x}_0 \in \Omega$ drawn from a distribution with pdf q, i.e. $\boldsymbol{x}_0 \sim q$. Here Ω is a generic space where data is encoded, we will see in what follows that in our case it will be simply a high-dimensional euclidean space that will represent the tensorial representation of the time series. Now suppose that we start to *gradually* corrupt $\boldsymbol{x}_0$ by adding increasing amounts of Gaussian noise until it becomes indistinguishable from pure white noise. Diffusion models learn to reverse this noising process and, once trained, they can generate new samples from an approximation of the original data distribution by iteratively denoising a sample of white noise.

In this section, we briefly review some of the basic mechanisms for the training of a diffusion model: the forward noising process, the reverse denoising process and its parametrization, the training objective, and the sampling routine. This will allow us to fix the notation (most of which is taken from [9]) that we will use in the reminder of the paper. First of all let us agree that throughout the manuscript, boldface indicates tensors, and plain symbols scalar value, for instance $\boldsymbol{x}$ and $\boldsymbol{\varepsilon}$ are tensors while α_t and β_t are scalar values.

Forward process This is the process that takes a sample from q and adds gaussian noise. In particular noisy data are obtained through a Markovian forward process over T diffusion steps

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}),$$

where the noise coefficients β_t determine the amount of noise added at each step. Using induction and the additivity of Gaussians, we have the following closed-form expression for the distribution after $t \leq T$ noising steps

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\mathbf{I}), \text{ with } \alpha_t := \prod_{i=1}^{t}(1-\beta_i), \quad (1)$$

so we can draw $\boldsymbol{x}_t$ directly by sampling $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$ and then using

$$\boldsymbol{x}_t = \sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\varepsilon}. \quad (2)$$

By choosing a sufficiently large T and an appropriate noise schedule [23] $\beta_1, \ldots, \beta_T$, $\boldsymbol{x}_T$ is nearly indistinguishable from white noise: $\boldsymbol{x}_T \sim \mathcal{N}(0, \mathbf{I})$.

Denoising process The backward process, or *denoising*, is on the other hand defined by the distribution $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ that regulates how noise is removed from the samples. While its direct computation is intractable, it can be approximated by a Gaussian process [9]

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t \mathbf{I}), \quad (3)$$

where the mean is parametrized by a *denoiser* neural network $\boldsymbol{\epsilon}_\theta$ with parameters θ:

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) \quad (4)$$

and

$$\sigma_t = \begin{cases} \frac{1-\alpha_{t-1}}{1-\alpha_t}\beta_t & \text{if } t > 1 \\ \beta_1 & \text{if } t = 1. \end{cases}$$

Training objective Maximizing the likelihood

$$p_\theta(\boldsymbol{x}_0) = \int_{\Omega^T} p_\theta(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)\, d\boldsymbol{x}_1 \cdots d\boldsymbol{x}_T \quad (5)$$

is unfeasible due to its computational intractability. Instead, training is achieved by *minimizing*, via stochastic gradient descent (SGD), a tractable negative evidence lower bound (ELBO) [9]

$$\mathcal{L}(\theta) := E_{\boldsymbol{x}_0, \boldsymbol{\varepsilon}, t}\left\|\boldsymbol{\epsilon}_\theta\left(\sqrt{\alpha_t}\,\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\varepsilon}_t,\, t\right) - \boldsymbol{\varepsilon}_t\right\|^2, \quad (6)$$

where $\boldsymbol{x}_0 \sim q$ is sampled from the original data distribution, $\boldsymbol{\varepsilon}_t \sim \mathcal{N}(0, \mathbf{I})$ is the added noise and t is sampled from the uniform distribution over the set $\{1, \ldots, T\}$.

Intuitively, we minimize the MSE between the predicted and the true added noise.

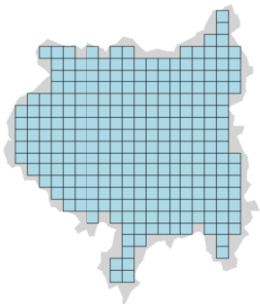

Figure 2: Spatial tiling of the Paris urban area into 267 patches, each consisting of 16×16 cells.

Sampling Generation starts by sampling a data point from white noise $\boldsymbol{x}_T \sim \mathcal{N}(0, \mathbf{I})$. Then, we can compute the mean $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ from the trained denoiser as in Eq. (4) and sample $\boldsymbol{x}_{t-1}$ from the distribution in Eq. (3). At the end of the denoising process, a final sample $\boldsymbol{x}_0$ is obtained, resulting in a generated sample that belongs to an approximation of the original data distribution.

Conditional generation If contextual information $\mathcal{C}$ (e.g., metadata, labels, spatial coordinates) is available, the denoiser can be extended by a conditional version $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \mathcal{C})$ to produce samples that better fit the conditional distributions $q(\boldsymbol{x} \mid \mathcal{C})$. The forward process is unchanged.

4 Denoiser Architecture for Spatio-Temporal Data

As we recalled in Section 3 the diffusion backward process crucially rely on the estimation of a parametric denoiser function $\boldsymbol{\epsilon}_\theta$ that in our case is a NN. In this section we describe in detail the architecture of this neural network exploiting the particular structure of the data we want to generate.

We build our architecture on DIFFWAVE [13] because its design aligns with the demands of our data: developed for audio synthesis, DIFFWAVE relies on bidirectional, dilated residual convolutions that can capture long-range temporal structure in fixed-length sequences, exactly the kind of temporal expressiveness we require. At the same time, its backbone is purely convolutional, avoiding the memory overhead of attention and lending itself to a straightforward generalization to the spatio-temporal setting.

We define *spatio-temporal patches* as a particular kind of temporal data, consisting of multivariate time series distributed over a regular, rectangular spatial grid. Our target objects are precisely these spatio-temporal patches, which represent localized regions of urban activity across both space and time. Formally, the data lives in the space

$$\Omega \equiv \mathbb{R}^{D \times T \times H \times W}.$$

Here D is the number of channels, T is the length of the time window, and H, W are, respectively, the height and width of the patch. Spatio-temporal patches can be used to tessellate a grid-like spatial domain (see Figure 2 for an example). To tackle the generative task, we adopt a *divide et impera* approach: instead of modeling the entire urban grid at once, we train models on individual patches, capturing localized spatio-temporal dynamics in a scalable manner.

DIFFWAVE's convolutional backbone can be lifted from one dimension to three with minimal changes, exactly what we need for the spatio-temporal patches considered here. By generalising the original 1D kernels to 3D (or to a hybrid 3D + 1D configuration), we can retain DIFFWAVE 's proven temporal modeling capability while directly modeling local spatial interactions within each patch. The rest of this section describes the resulting denoiser architecture.

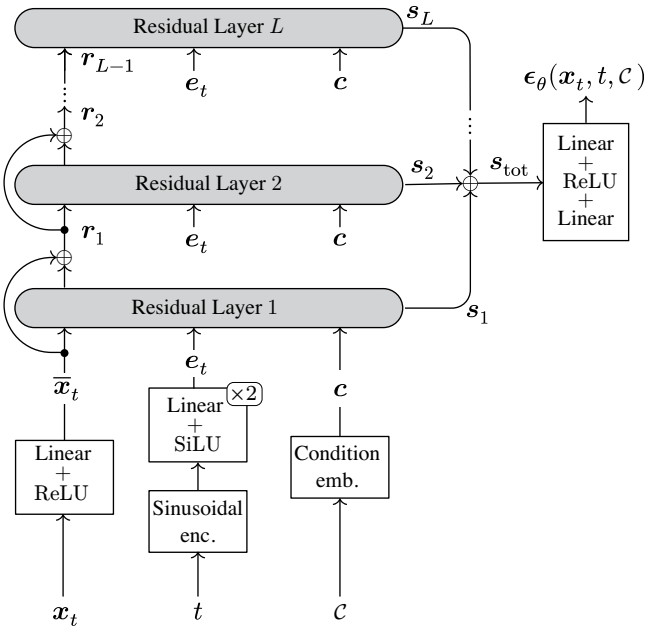

Figure 3: The denoiser architecture in DIFFWAVE. At each training or inference step, the model receives a noisy sample $\boldsymbol{x}_t$, a diffusion timestep t, and a set of contextual features $\mathcal{C}$. The sample $\boldsymbol{x}_t$ is linearly projected with ReLU, the timestep t is encoded via sinusoidal and linear layers into $\boldsymbol{e}_t$, and the contextual features $\mathcal{C}$ are embedded into $\boldsymbol{c}$. These encoded inputs are then passed to a stack of residual layers. The skip connections from all residual layers are summed and passed through a final MLP to predict the denoised output $\epsilon_\theta(\boldsymbol{x}_t, t, \mathcal{C})$.

In spatio-temporal data, two additional spatial dimensions (height and width) are present alongside time. To jointly model patterns across all three dimensions, we extend the bidirectional dilated convolutional block into a dedicated *spatio-temporal module*. The next section describes two variants of this module and compares them to vanilla DIFFWAVE and the attention-based CSDI [27].

Around this spatio-temporal module there are three auxiliary modules: an input-embedding layer, a diffusion-step encoder and a condition encoder. Indeed, the denoiser takes three inputs:

1. a noisy spatio-temporal patch $\boldsymbol{x}_t$, obtained during training by corrupting a clean sample $\boldsymbol{x}_0$ according to Eq. (2), and during inference by iteratively denoising $\boldsymbol{x}_{t+1}$;
2. the current diffusion step t;
3. a set of metadata $\mathcal{C}$ that contains cell coordinates and any other contextual information linked to the patch (e.g., day of collection, weather conditions, sensor type).

The triplet $(\boldsymbol{x}_t, t, \mathcal{C})$ is encoded into $(\overline{\boldsymbol{x}}_t, \boldsymbol{e}_t, \boldsymbol{c})$ by the above mentioned auxiliary modules and then fed to the network.

Below, we provide a detailed description of each component of the denoiser. See Figure 3 for an overview of the architecture.

Input Embedding. The input tensor $\boldsymbol{x}_t$ is projected into $\overline{D}$ feature channels via a 1×1 convolution, which applies a shared linear transformation across all spatio-temporal positions, followed by a ReLU activation

$$\overline{\boldsymbol{x}}_t = \mathrm{ReLU}\big(\mathrm{Conv1x1}(\boldsymbol{x}_t)\big) \in \mathbb{R}^{\overline{D} \times T \times H \times W}.$$

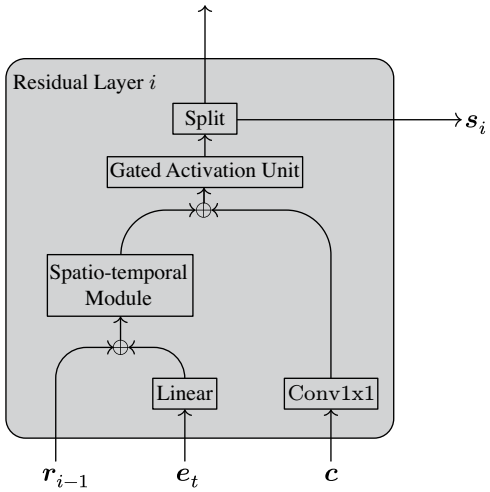

Figure 4: General scheme of a residual layer. Each layer receives the previous residual input $\boldsymbol{r}_{i-1}$, the encoded diffusion step $\boldsymbol{e}_t$, and the embedded conditions $\boldsymbol{c}$. These inputs are linearly projected and combined, then passed through a spatio-temporal module. The output flows through a gated activation unit and is split to produce a residual update and a skip connection $\boldsymbol{s}_i$. This structure accommodates all four spatio-temporal module variants evaluated in this work.

Diffusion Step Embedding. The diffusion step t is encoded into a 128-dimensional vector using sinusoidal encoding [29]

$$\boldsymbol{e}_t = \Bigg(\sin\left(10^{\frac{0\times4}{63}} t\right), \cdots, \sin\left(10^{\frac{63\times4}{63}} t\right), \cos\left(10^{\frac{0\times4}{63}} t\right), \cdots, \cos\left(10^{\frac{63\times4}{63}} t\right) \Bigg), \tag{7}$$

then passed through two linear layers to produce the final embedding, still denoted $\boldsymbol{e}_t \in \mathbb{R}^{128}$.

Condition Embedding. The denoiser can exploit a heterogeneous set of metadata $\mathcal{C}$ associated with each patch. We encode every item in $\mathcal{C}$ into a vector and then stack the results into a single tensor that matches the spatio-temporal layout of $\boldsymbol{x}_t$. Specifically:

- *Categorical features:* (e.g., day of the week, sensor type) one-hot encoded and projected via a linear layer.
- *Cell positions:* each spatial coordinate (row and column) is encoded independently with a sinusoidal positional vector, using the same formulation as Eq. (7). The fixed sine–cosine basis preserves linear offsets between positions, allowing the network to reason about both absolute and relative positions on the grid [29].
- *Timestep positions:* when temporal self-attention is used (e.g., CSDI), we add positional encodings for each timestep, also via Eq. (7). Since attention mechanisms are inherently order-agnostic, this positional information is necessary to allow the model to capture the relative or absolute order of elements in the sequence [29].

Each resulting vector is broadcast along the missing spatial or temporal dimensions (e.g., cell positions are time-invariant, while weather descriptors may be space-invariant) so that every code becomes a four-dimensional tensor. Concatenating these tensors along the channel axis yields $\boldsymbol{c} \in \mathbb{R}^{K\times T\times H\times W}$, where K is the total number of metadata channels after projection. The condition tensor $\boldsymbol{c}$ is supplied to every residual block with the timestep embedding $\boldsymbol{e}_t$.

Residual Layers. The denoiser comprises L residual layers, indexed by $i = 1, \ldots, L$, each designed to clean the noisy input by modeling temporal and spatial context through a spatio-temporal module. Below, we describe the operations carried out inside each layer. See Figure 4 for an overview of the residual block.

Let $\boldsymbol{r}_{i-1} \in \mathbb{R}^{\overline{D}\times T\times H\times W}$ be the input to the i-th residual block (with $\boldsymbol{r}_0 = \overline{\boldsymbol{x}}_t$), and let $\boldsymbol{e}_t$ and $\boldsymbol{c}$ be the diffusion step and condition embeddings, respectively.

Each residual layer first projects the timestep embedding via a linear transformation: $\bar{\boldsymbol{e}}_t = \mathrm{Linear}_{\mathrm{time}}(\boldsymbol{e}_t)$, where $\mathrm{Linear}_{\mathrm{time}}\colon \mathbb{R}^{128} \to \mathbb{R}^{\overline{D}}$ which is then broadcast across the full spatio-temporal shape.

The residual computation proceeds as follows:

$$\boldsymbol{y} = \mathrm{STModule}\big(\boldsymbol{r}_{i-1} + \bar{\boldsymbol{e}}_t\big) + \mathrm{Conv1x1}_{\mathrm{cond}}(\boldsymbol{c}),$$

where $\mathrm{Conv1x1}_{\mathrm{cond}} : \mathbb{R}^{K\times T\times H\times W} \to \mathbb{R}^{\overline{D}\times T\times H\times W}$ acts as shared linear transformation on the channel dimension, shared between all spatio-temporal positions.

The result $\boldsymbol{y}$ is then split along the channel dimension into two halves, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, and passed through a gated activation unit $\boldsymbol{g} = \sigma(\boldsymbol{y}_1) \odot \tanh(\boldsymbol{y}_2)$, followed by another linear layer and a final split that produces the residual update $\boldsymbol{r}'_i$ and the skip connection $\boldsymbol{s}_i$. The layer output is: $\boldsymbol{r}_i = \boldsymbol{r}_{i-1} + \boldsymbol{r}'_i$.

The skip connections from all L layers are summed to form $\boldsymbol{s}_{\mathrm{tot}} = \sum_{i=1}^{L} \boldsymbol{s}_i$, which is passed through a two-layer MLP to produce the predicted noise $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t, \mathcal{C})$.

5 Spatio-temporal modules

The spatio-temporal module is the core architectural component responsible for capturing spatio-temporal dependencies in the denoiser. It defines the operations applied at the center of each residual block and is the key differentiator among the variants evaluated in this work.

As a first spatio-temporal module, we consider the attention-based approach used in CSDI [27], originally proposed for multivariate time series imputation. Like our models, CSDI builds on a DiffWave-style residual backbone, but replaces its convolutional blocks with a pair of attention layers: a temporal self-attention layer followed by a *feature* self-attention layer. Treating the reading at each location as a separate feature allows the latter layer to act as spatial attention. Although attention offers global context, its quadratic cost in sequence length makes it expensive on the large spatio-temporal patches. Moreover, the original DiffWave already provides an efficient mechanism for long-range *temporal* context through dilated 1D convolutions. We therefore revisit a purely convolutional solution that can be extended naturally to three dimensions. Our key idea is to replace the 1D bidirectional dilated convolutions of DiffWave with 3D counterparts that operate simultaneously over time, height, and width. We designed two 3-dimensional variants of DiffWave, named DiffWave3D and DiffWave3D1D, and compared them with the baselines DiffWave and CSDI. The characteristics of all these four variants are described below:

DiffWave Mirrors the original DiffWave. To accommodate spatio-temporal patches as input, we employ 3D bilateral dilated convolutions with a kernel size of $(3, 1, 1)$, where the first dimension corresponds to the temporal axis and the remaining two to the spatial dimensions. This is functionally equivalent to using 1D bilateral dilated convolutions with a kernel size of 3 along the temporal dimension. A dilation factor 2^i along the temporal axis is applied, i being the index of the residual layer. In this case, each spatial cell is processed independently and no spatial dependencies are modeled.

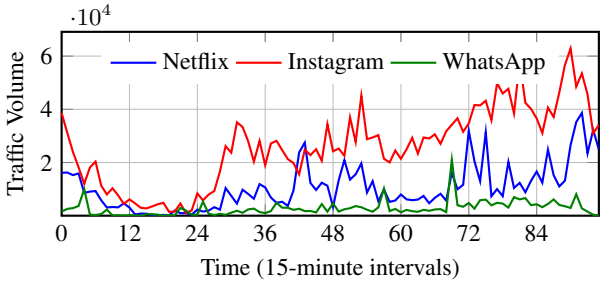

Figure 5: Time series from the NETMOB23 dataset for three services in a randomly selected cell and day.

DIFFWAVE3D Employs full 3D convolutions with kernel size $(3, 3, 3)$, where dilation 2^i applies along the temporal axis only modeling local spatial patterns alongside temporal structure.

DIFFWAVE3D1D This variant is a mix between the previous two. The idea is to account both for local temporal patterns and spatial autocorrelation. This spatio-temporal module splits the C channels into two halves: one half goes through a $(3, 1, 1)$ dilated temporal convolution (as in DIFFWAVE), the other through a $(3, 3, 3)$ dilated spatio-temporal convolution (as in DIFFWAVE3D). This choice allows for simultaneous modeling of temporal dependencies and spatial autocorrelation within the same residual block. By preserving the dedicated temporal path of DIFFWAVE, DIFFWAVE3D1D is able to model local temporal correlations more precisely than DIFFWAVE3D, while still retaining the ability to capture spatial patterns through its 3D convolutional branch.

CSDI Block Employs the CSDI attention mechanism [27]: Temporal and spatial self-attention layers are applied, with spatial dimensions flattened and positional encodings added along the other conditions.

6 Experiments and Evaluation

In this section, we present our experimental setup and evaluation methods, detailing the dataset, model configurations, and the metrics used to assess generation quality and spatial coherence. All code used for training and evaluating the models is available at [19].

The NETMOB23 *dataset*, released during the NETMOB23 Data Challenge [21], provides aggregated upload and download volumes from the Orange 4G mobile network, collected across 20 French urban areas. Traffic is categorized into 68 widely-used mobile services (e.g., Instagram, YouTube, Gmail). Each area is partitioned into a uniform grid of 100m $\times$ 100m cells, with data sampled at 15-minute intervals—offering high spatial and temporal granularity. The full dataset spans 76 consecutive days, from March 16 to May 31, 2019.

For our experiments, we focus on the city of Paris and three selected services: Netflix, WhatsApp, and Instagram. These services were chosen due to the large dataset size (approximately 3.8GB per service for Paris) and the diversity of user behavior they represent. While limited in number, they provide a compact yet representative snapshot of mobile activity in the urban area. Figure 5 shows a daily record of the activity on these three services on a given cell.

Following the patch-based approach introduced in Section 4, we tessellate the Paris grid into 267 non-overlapping patches of 16 $\times$ 16 cells (see Figure 2). Due to the irregular urban footprint, some peripheral cells fall outside these patches and are excluded from the analysis. Each time series is segmented into daily sequences of 96 timesteps, resulting in 76 daily samples per patch. We treat each of the three selected services (Netflix, WhatsApp, and Instagram) as a separate univariate time series. As a result, our dataset comprises $3 \times 76 \times 267$ spatio-temporal samples, each represented as a tensor of shape $(1, 96, 16, 16)$.

Raw traffic volumes are strongly right-skewed. Following common practice [15], we apply a $\ln(1 + x)$ transformation componentwise to each spatio-temporal patch to reduce skewness and then rescale the transformed values linearly to the interval $[-1, 1]$ to match the input range expected by our neural networks.

Each sample is associated with the following metadata. *i* Grid position: a $(2, 16, 16)$ tensor containing the row and column indices of each cell. *ii* Day of week: a categorical variable with values Monday-Sunday. *iii* One of Netflix, WhatsApp, or Instagram.

Metrics Despite a growing number of generative models, the field lacks consensus on what constitutes high-quality synthetic time series and how to quantify it [26]. Our task is even less addressed by standard metrics, as we aim to model not only temporal dynamics but also spatial dependencies and conditional agreement with metadata. In light of this, we adopt complementary criteria to evaluate our method from four distinct perspectives: visual fidelity, conditional distribution matching, spatial coherence, and downstream utility.

While our generative models are trained on spatio-temporal patches, we discard the patch structure during evaluation and assess the quality of the generated data at the individual cell level. This enables finer-grained analysis and ensures a fair comparison with baseline methods. At this stage, each sample corresponds to a univariate time series of length 96, along with its associated metadata: a pair specifying the cell position, the day, and the service.

Visual inspection. For a given cell, service and day of week, we visually compare real samples to synthetic ones generated with the different denoiser architectures. We also visually inspect the spatial coherence of the temporal data.

Wasserstein distance. To evaluate the *conditional agreement* between real and synthetic data, we compute empirical distributions of real and synthetic data conditioned on cell position p, service s, and a binary weekend indicator d. Let $Q^R_{p,s,d}$ and $Q^G_{p,s,d}$ denote the empirical distributions from respectively the real and generated data, conditioned on the attributes p, s, d. The Wasserstein distance provides a meaningful way to quantify how close the model-generated distribution is to the real one [3], and it naturally extends to the conditional setting. It is given by

$$W\left(Q^R_{p,s,d},\, Q^G_{p,s,d}\right) = \inf_{\pi \in \mathfrak{S}_n} \left(\frac{1}{96} \sum_{i=1}^{n} \left\| \boldsymbol{x}_i - \boldsymbol{y}_{\pi(i)} \right\|^2 \right)^{\frac{1}{2}},$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{96}$ for $i = 1, \dots n$ are respectively the real and generated samples with attribute p, s, d, and $\mathfrak{S}_n$ is the set of permutations of n elements. We finally compute the average Wasserstein distance over all (p, s, d) combinations:

$$\frac{1}{N} \sum_{p,s,d} W\left(Q^R_{p,s,d},\, Q^G_{p,s,d}\right), \tag{8}$$

where N is the number of unique (p, s, d) tuples.

Spatial autocorrelation measure. Given two time series $\boldsymbol{x} = (x_1, \dots, x_{96})$ and $\boldsymbol{Y} = (y_1, \dots, y_{96})$, we compute the Pearson correlation coefficient:

$$r_{\boldsymbol{xy}} = \frac{\sum_{i=1}^{96} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{96} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{96} (y_i - \bar{y})^2}}. \tag{9}$$

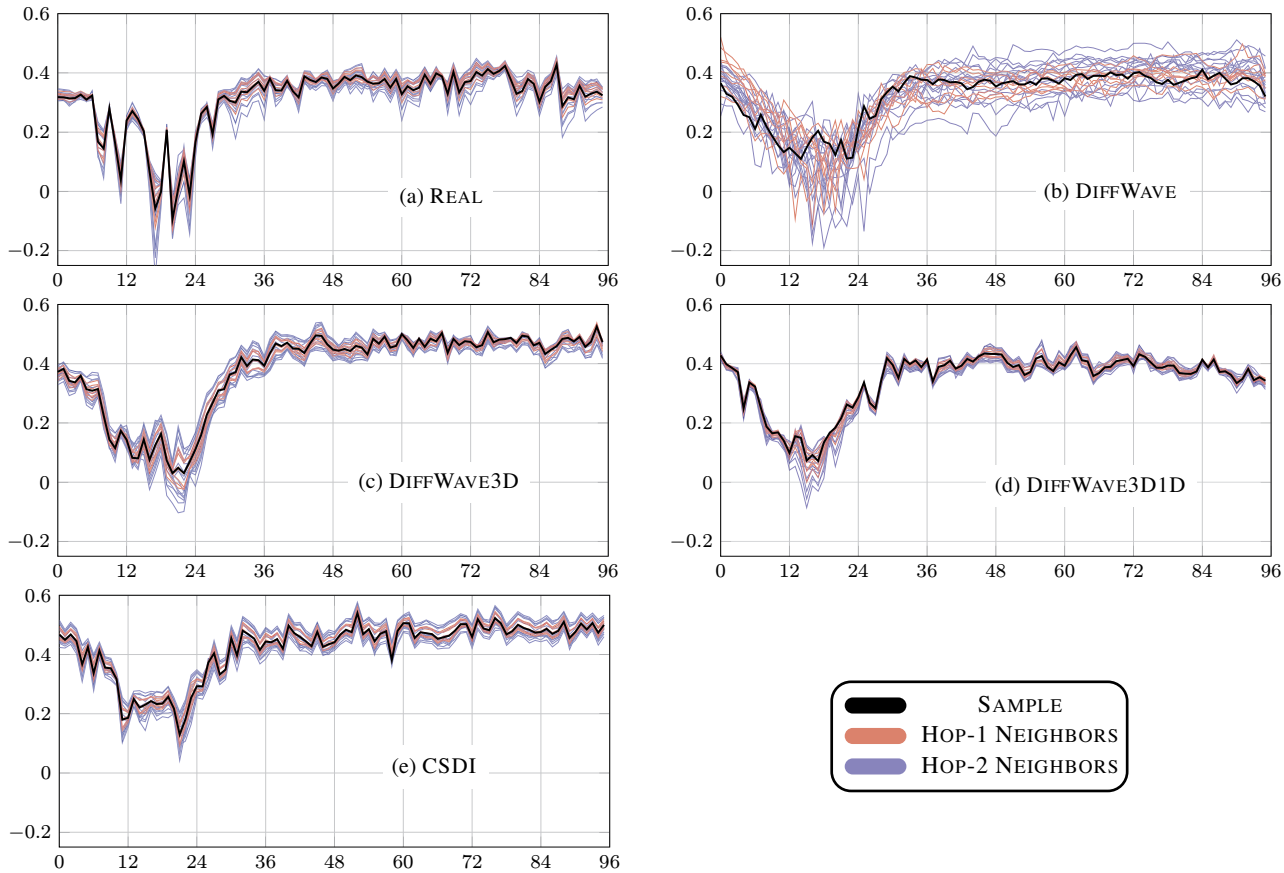

Figure 6: Comparison of a sampled time series in the normalized domain and its spatial neighbors. Panel (a) shows the real data, while panels (b)–(e) show time series generated by DIFFWAVE, DIFFWAVE3D, DIFFWAVE3D1D, and CSDI, respectively. In each case, the generated data is conditioned on the same spatial position, service, and day of the week as the real sample. Lines represent the sampled cell (black), its hop-1 neighbors (green), and hop-2 neighbors (blue). The vanilla DIFFWAVE (b) captures the overall daily trend of the central time series, but fails to reproduce coherent patterns in the surrounding cells. The hop-1 and hop-2 neighbor trajectories drift apart, revealing a lack of spatial autocorrelation.

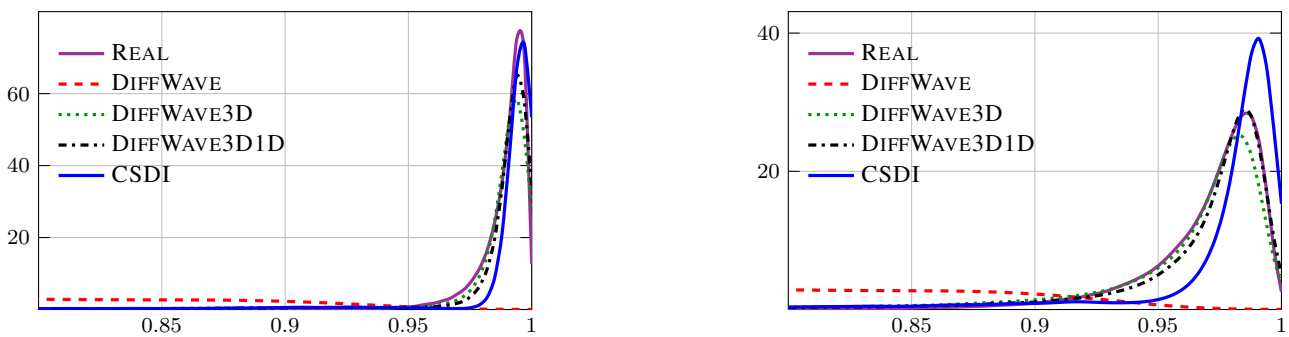

Figure 7: Distribution of average Pearson correlation coefficients between each cell and its 1-hop (left) and 2-hop (right) neighbors.

For each time series $\boldsymbol{x}$, we define $\mathcal{R}_1(\boldsymbol{x})$ as the set of first-order (hop-1) neighboring time series—i.e., the 8 surrounding cells on the same day—and $\mathcal{R}_2(\boldsymbol{x})$ as the set of second-order (hop-2) neighbors (up to 16 cells). We compute:

$$\mathcal{P}_1(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\mathcal{R}_1(\boldsymbol{x})} \frac{r_{\boldsymbol{xy}}}{|\mathcal{R}_1(\boldsymbol{x})|}, \quad \mathcal{P}_2(\boldsymbol{x}) = \sum_{\boldsymbol{y}\in\mathcal{R}_2(\boldsymbol{x})} \frac{r_{\boldsymbol{xy}}}{|\mathcal{R}_2(\boldsymbol{x})|}. \quad (10)$$

We compare the distributions of $\mathcal{P}_1$ and $\mathcal{P}_2$ for real and synthetic data to evaluate the preservation of spatial autocorrelation.

Train-on-synthetic, test-on-real (TSTR). This metric, introduced in [7], evaluates the utility of synthetic data for downstream tasks. It is applicable whenever a supervised task can be defined in the domain of the real and generated data. We implement it by training a classifier (either an MLP or a CNN) on synthetic samples and evaluating its performance on real data for the task of service classification.

Experimental Evaluation We trained all models for 100 epochs using the Adam optimizer with a learning rate of 2×10^{-4} and a weight decay of 1×10^{-4}. The diffusion process was defined over $T = 50$ timesteps (as one of the best performing values tested in [13]), with the cosine noise schedule from [23]

Table 1: Performance of models (Wasserstein distance)

MODEL	WASS. DIST.	#PARAMS	TRAIN TIME
CSDI	1.7105 ± 0.0711	242,849	16h 50m
DIFFWAVE	1.2033 ± 0.0090	220,097	1h 45m
DIFFWAVE3D	1.2129 ± 0.0059	711,617	3h 00m
DIFFWAVE3D1D	$\mathbf{1.2027 \pm 0.0140}$	**921,057**	**4h 30m**

$\alpha_t = \cos^2\left((t/T + \varepsilon)/(1+\varepsilon) \cdot \pi/2\right)$. All DIFFWAVE-based models shared the same architecture scheme: five residual layers with 32 residual channels and a dilation cycle length of 5—except for DIFFWAVE3D1D, which used 64 residual channels to accomodate its hybrid architecture. The CSDI model employed a transformer-based structure with 4 attention heads, 32 residual channels, and 4 residual layers. All models used sequences of length 96 with a single input channel. Training used exponential moving average (EMA) updates every 10 steps (decay rate: 0.995). All experiments were conducted on a machine equipped with 4 NVIDIA A10G GPUs, each with 22.4 GiB of memory. To prevent data leakage, the dataset consisting of spatio-temporal patches described at the beginning of this section was randomly split into a 70% training set and a 30% test set. After training, synthetic spatio-temporal patches were generated using the same metadata (i.e., patch location, service, and day) as those in the test set. All evaluations were conducted exclusively on the test set, comparing the real 30% subset and its synthetic counterparts generated by the four methods described in the paper. As previously anticipated we discard the patch structure at evaluation time and compute all the evaluation metrics at the individual cell level. All reported metrics are averaged over three different random splits, with standard deviations provided to assess variability.

We begin with a visual inspection of the generated data in Figure 6, where we plot a real time series sample alongside its hop-1 and hop-2 neighbors. In the same Figure, we show the corresponding generated samples, conditioned on the same position, service, and day of the week. The results highlight the importance of incorporating spatial relationships in the denoising process: although the fully local DIFFWAVE model captures the overall daily trend of the central cell, its neighboring time series drift apart, indicating weak spatial coherence.

Wasserstein distances are shown in Table 1. We observe a substantial equivalence between the DIFFWAVE-based methods and a lower performance of CSDI that also requires a larger training time, as indicated in the last column. While DIFFWAVE3D shows a slight degradation compared to the fully 1D DIFFWAVE model, the hybrid DIFFWAVE3D1D architecture recovers this margin and achieves the best overall score. This suggests that combining 1D temporal convolutions with 3D spatio-temporal convolutions enhances local generative fidelity while preserving spatial coherence.

For each combination of day and service, we randomly sampled 1,000 cells and computed the average Pearson correlation coefficient between each cell and its hop-1 and hop-2 neighbors. These averages were used to form the distributions of spatial autocorrelation across the dataset. To measure the discrepancy between the real autocorrelation distribution and that of each tested generative model, we computed their Wasserstein distance. The results are reported in Table 2. A more detailed view of the distribution of Pearson correlation coefficients across different generative methods is shown in Figure 7, supporting the visual impressions from Figure 6. From Figure 7, we also see that DIFFWAVE fails to adequately reproduce the real spatial autocorrelation, whereas while CSDI overestimates it. DIFFWAVE3D and DIFFWAVE3D1D demonstrate the best alignment with the real distribution, capturing spatial dependencies more accurately.

Table 2: Discrepancy of spatial autocorrelation from the real case (Wasserstein distance)

MODEL	HOP-1	HOP-2
CSDI	0.0107	0.0151
DIFFWAVE	0.2916	0.2714
DIFFWAVE3D	0.0080	0.0136
DIFFWAVE3D1D	**0.0078**	**0.0091**

To compute TSTR accuracy, we trained two classifiers, one based on fully connected layers and the other on 1D convolutional layers. We used a learning rate of $1e{-}4$ and early stopping with patience 5. The fully connected model includes three linear layers with ReLU activations (29,315 parameters), while the convolutional model combines two $\mathrm{Conv1D}$ layers, ReLU activations, adaptive average pooling, and a final linear layer (6,531 parameters). Results are reported in Table 3, showing that classifiers trained on data generated by the 3-dimensional DIFFWAVE variants achieve the best performance.

Table 3: Average accuracy for each model and classification method.

METHOD	MODEL	ACCURACY
MLP	DIFFWAVE	0.9058 ± 0.0143
	DIFFWAVE3D	0.9149 ± 0.0002
	DIFFWAVE3D1D	$\mathbf{0.9270 \pm 0.0132}$
	CSDI	0.8999 ± 0.0074
CNN	DIFFWAVE	0.8561 ± 0.0135
	DIFFWAVE3D	0.8959 ± 0.0060
	DIFFWAVE3D1D	$\mathbf{0.9115 \pm 0.0121}$
	CSDI	0.8836 ± 0.0099

7 Conclusions

This work addresses the gap in modeling spatial autocorrelation when generating temporal data through diffusion models. We carried out all experiments on a novel, real-world dataset, offering a fresh perspective on how current generative methods perform in complex urban scenarios. Our proposed solution demonstrates competitive performance in capturing the underlying distribution of spatio-temporal data, as measured by the Wasserstein distance, while more effectively reproducing realistic spatial autocorrelation. This improvement also translates into practical utility, as evidenced by the performance on the TSTR (Train on Synthetic, Test on Real) task.

Limitations and future work The current version of DIFFWAVE3D1D is limited to grid-structured spatial data. We believe that the core idea of incorporating spatial relationships into a diffusion-based generative framework can be extended to irregular spatial domains using graph neural networks.

The applicability of the current implementation of our method is limited by the amount of available metadata intrinsically linked to the temporal data. On a more applied level, we are interested in advancing spatio-temporal data generation in urban contexts. In particular, we aim to develop models capable of generating data conditioned on contextual urban information, such as land use, demographics, or infrastructure, in a zero-shot or few-shot setting, enabling broader generalization to unseen cities, and mobile services. Finding the right set of urban descriptors that allow for a semantically meaningful control over the generation process remains an open challenge in our framework, which we aim to address in future work. Finally, we intend to apply DIFFWAVE3D1D to other spatio-temporal domains, such as environmental monitoring, transportation networks, or epidemiological data.

Acknowledgements

This research was performed using data made available by Orange within the NETMOB 2023 Data Challenge [21].

This research was supported by the "Resilienza Economica e Digitale" project (CUP D67G23000060001) funded by the Italian Ministry of University and Research (MUR) as "Department of Excellence" (Dipartimenti di Eccellenza 2023-2027, Ministerial Decree no. 230/2022)

References

[1] J. L. Alcaraz and N. Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*.

[2] J. M. L. Alcaraz and N. Strodthoff. Diffusion-based conditional ecg generation with structured state space models. *Computers in biology and medicine*, 163:107115, 2023.

[3] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[4] A. Cini, I. Marisca, and C. Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations*.

[5] A. Coletta, S. Gopalakrishnan, D. Borrajo, and S. Vyetrenko. On the constrained time-series generation problem. *Advances in Neural Information Processing Systems*, 36:61048–61059, 2023.

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

[7] C. Esteban, S. L. Hyland, and G. Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

[8] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

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

[10] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

[11] J. Jeon, J. Kim, H. Song, S. Cho, and N. Park. Gt-gan: General purpose time series synthesis with generative adversarial networks. *Advances in Neural Information Processing Systems*, 35:36999–37010, 2022.

[12] M. Jin, H. Y. Koh, Q. Wen, D. Zambon, C. Alippi, G. I. Webb, I. King, and S. Pan. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

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

[14] Y. Kossale, M. Airaj, and A. Darouichi. Mode collapse in generative adversarial networks: An overview. In *2022 8th International Conference on Optimization and Applications (ICOA)*, pages 1–6. IEEE, 2022.

[15] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer, New York, NY, 2013. ISBN 978-1-4614-6849-3. URL https://doi.org/10.1007/978-1-4614-6849-3. See Chapter 3: Data Pre-Processing.

[16] X. Li, V. Metsis, H. Wang, and A. H. H. Ngu. Tts-gan: A transformer-based time-series generative adversarial network. In *International conference on artificial intelligence in medicine*, pages 133–143. Springer, 2022.

[17] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[18] M. Liu, H. Huang, H. Feng, L. Sun, B. Du, and Y. Fu. Pristi: A conditional diffusion framework for spatiotemporal imputation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1927–1939. IEEE Computer Society, 2023.

[19] G. Loddi. Code for "A conditional generative diffusion model for spatio-temporal data". https://github.com/g-loddi/DiffWave3D1D, 2025. GitHub repository.

[20] I. Marisca, C. Alippi, and F. M. Bianchi. Graph-based forecasting with missing data through spatiotemporal downsampling. *arXiv preprint arXiv:2402.10634*, 2024.

[21] O. E. Martínez-Durive, S. Mishra, C. Ziemlicki, S. Rubrichi, Z. Smoreda, and M. Fiore. The netmob23 dataset: A high-resolution multi-region service-level mobile data traffic cartography. *arXiv preprint arXiv:2305.06933*, 2023.

[22] S. S. Narasimhan, S. Agarwal, O. Akcin, S. Sanghavi, and S. P. Chinchali. Time weaver: A conditional time series generation model. In *International Conference on Machine Learning*, pages 37293–37320. PMLR, 2024.

[23] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[24] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pages 8857–8868. PMLR, 2021.

[25] L. Shen, W. Chen, and J. Kwok. Multi-resolution diffusion models for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[26] M. Stenger, R. Leppich, I. Foster, S. Kounev, and A. Bauer. Evaluation is key: a survey on evaluation measures for synthetic time series. *Journal of Big Data*, 11(1):66, 2024.

[27] Y. Tashiro, J. Song, Y. Song, and S. Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34:24804–24816, 2021.

[28] H. Thanh-Tung and T. Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Y. Yang, M. Jin, H. Wen, C. Zhang, Y. Liang, L. Ma, Y. Wang, C. Liu, B. Yang, Z. Xu, et al. A survey on diffusion models for time series and spatio-temporal data. *arXiv preprint arXiv:2404.18886*, 2024.

[31] X. Yi, Y. Zheng, J. Zhang, and T. Li. St-mvl: filling missing values in geo-sensory time series data. In *Proceedings of the 25th international joint conference on artificial intelligence*, 2016.

[32] J. Yoon, D. Jarrett, and M. Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.