A synthetic dataset for Time Series Super-Resolution with Deep Learning

Julio Ibarra-Fiallo[1], Juan A. Lara[2], and D'hamar Agudelo-Moreno[1]

[1]Colegio de Ciencias e Ingenierías, Universidad San Francisco de Quito, Cumbayá, Ecuador
[2]Universidad de Córdoba, Córdoba, España
*corresponding author: Julio Ibarra-Fiallo (jibarra@usfq.edu.ec)

ABSTRACT

The increasing application of time-series analysis in fields like biomedical engineering, telecommunications, and industrial monitoring emphasizes the need for high-quality data to train and evaluate advanced machine learning models. Acquiring real-world temporal data at suitable resolutions is often limited by ethical, economic, or practical constraints. To address this, we introduce CoSiBD (Complex Signal Benchmark Dataset for Super-Resolution), a synthetic dataset of complex temporal signals designed for training and assessing AI models, particularly deep learning systems, in tasks like temporal super-resolution and signal processing. CoSiBD comprises 2,500 high-resolution signals ($N = 5{,}000$ samples each over a reference domain $\tau \in [0, 4\pi]$) with corresponding low-resolution versions at four levels (150, 250, 500, and 1,000 samples) obtained via simple uniform decimation (uniform subsampling) of the original sequence. Each signal is provided in three formats (NumPy arrays, plain text, and JSON) with comprehensive metadata documenting all generation parameters, including random seeds for full reproducibility. CoSiBD includes diverse signals with non-uniform frequency modulations, capturing gradual transitions and abrupt high-frequency events to mirror real-world dynamics, and provides both clean and noisy variants. We report a technical validation focusing on spectral consistency across sampling rates and noise to support training and evaluation.

Background & Summary

The analysis and simulation of temporal signals are fundamental across science and engineering. These techniques provide critical insights into dynamic processes in multiple domains. In biomedical research[1], electroencephalography (EEG) and electrocardiography (ECG) analyses reveal brain and heart function[2,3]. Telecommunications rely on signal processing to ensure data fidelity across noisy media[4], while finance uses time-series forecasting for risk and trend analysis[5]. Industrial monitoring detects equipment faults using temporal patterns[6], and environmental science applies similar techniques to climate and environmental monitoring using remote-sensor time series[7]. Developing robust tools for interpreting time-varying data continues to support both scientific discovery and practical applications.

Recent advances in deep learning have contributed significantly to this field by enabling automatic extraction of complex features from raw signals. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) units, and Generative Adversarial Networks (GANs) have demonstrated improved performance over traditional techniques in image, speech, and time-series processing tasks[8,9]. These models support fine-grained signal reconstruction and forecasting, allowing researchers to explore temporal dynamics in new ways.

Despite this progress, deep learning methods for temporal signal processing often require large quantities of labeled, high-quality data. Access to such data is frequently constrained by medical privacy regulations such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA)[10]. In other domains, including environmental monitoring using remote sensors and industrial monitoring, data availability is limited by practical and economic barriers to sensor deployment and data collection[5]. These limitations are particularly relevant in super-resolution (SR) tasks, where models require paired low- and high-resolution signals for effective training.

Temporal SR, which enhances resolution over time, has broad potential. In biomedical monitoring and sensing, SR can help reconstruct higher-resolution physiological time series (e.g., ECG/EEG), potentially improving the analysis of neural oscillations[2] and subtle physiological irregularities[3]. SR also applies to audio/speech enhancement, industrial vibration monitoring, and telecommunications, where higher temporal resolution can increase sensitivity to rapid changes and improve signal quality.

Traditional SR methods such as polynomial interpolation, frequency-domain transforms, and splines each have limitations.

Polynomial models are often insufficient for capturing nonlinear dynamics; frequency-domain methods are susceptible to noise[7]; and splines, though flexible, may not generalize well to complex signal variability[11,12]. Many of these methods also assume uniform partitioning, which may not align with the multi-scale, irregular structure of natural temporal phenomena.

Deep learning offers adaptive alternatives to these traditional methods. CNNs are capable of modeling spatio-temporal structure, RNNs and LSTMs capture long-range dependencies in time, and GANs can learn high-resolution representations through adversarial training[8,9]. While GANs have achieved strong results in image SR[13], their application to time-series SR remains relatively new. Preliminary work on synthetic time-series generation indicates potential[13,14], but the lack of accessible, high-quality paired datasets remains a significant barrier to progress.

Synthetic datasets offer one solution to this problem, allowing researchers to design reproducible training environments that reflect the structure and variability of real-world signals. Prior studies have used synthetic data in domains such as fluid dynamics[15], bioimaging[16], and live-cell imaging[17], demonstrating that synthetic approaches can help simulate complexity while avoiding legal and practical restrictions associated with real-world data.

To support research in super-resolution for time-series data, we present the Complex Signal Benchmark Dataset (CoSiBD). CoSiBD is a synthetic dataset composed of time-series signals with variable resolution, frequency characteristics, and noise levels. The dataset is intended to provide a resource for training and evaluating SR models under controlled, reproducible conditions. It includes non-stationary, piecewise-structured signals (via non-uniform interval partitioning with change-points), multiple levels of resolution and noise, a technical validation suite, and publicly available Python code to facilitate use. CoSiBD has been used in research presented at the International Conference on Signal Processing and Machine Learning[14] and is made available to support further development in deep learning approaches for temporal super-resolution.

To further position CoSiBD with respect to existing public synthetic time-series resources, we summarize representative datasets and simulators and highlight the practical gap addressed by our benchmark.

Related synthetic time-series resources

Publicly available synthetic resources for temporal signals exist, but they are typically designed for tasks other than time-series super-resolution (SR), or they target a specific domain. In wireless communications, the RadioML family provides large collections of synthetic complex I/Q sequences with varying SNR and channel impairments, mainly to benchmark automatic modulation classification rather than paired SR reconstruction[18–20]. In biomedical signal processing, physiological simulators such as ECGSYN (ECG) and SEREEGA (EEG) enable controlled generation with tunable morphology, sampling settings, and noise, supporting method development when real data access is constrained[21–23]. In power systems, LoadGAN provides multi-resolution generation of load time series across sampling rates and time horizons (from sub-second to long-term scales), but it is not distributed as a standardized paired SR benchmark[24]. Domain-specific paired low-/high-resolution training data can also be produced via physical forward modeling, e.g., low- and high-resolution 1D seismic traces for learning-based resolution enhancement[25].

Table 1 summarizes these representative resources and highlights a practical gap: while many tools provide synthetic signals, they usually do not jointly offer (i) multi-factor paired LR–HR signals for time-series SR, (ii) a clear pairing protocol for low-resolution observations aligned to reconstructing the original HR target (here implemented via simple uniform decimation), and (iii) per-signal metadata enabling deterministic regeneration and principled benchmarking. CoSiBD is designed to address this gap by providing multi-resolution paired signals, explicit nuisance modeling (noise and structured interference), and comprehensive metadata for reproducible SR benchmarking across multiple difficulty levels.

Methods

The methodology used to generate the synthetic temporal signals that constitute the CoSiBD dataset is illustrated in Figure 1. The process was designed to produce signals that reflect general characteristics of real-world temporal data, such as variable frequency content, continuous transitions, and intermittent high-frequency activity. A key aspect of the procedure is the ability to produce signals at different resolution levels, supporting the generation of paired datasets for evaluating super-resolution (SR) algorithms.

Design rationale inspired by real signals. To address the concern that the dataset is "too artificial", we derived the simulator degrees of freedom from qualitative observations across representative physiological (EEG/ECG) and speech signals. In particular, real signals exhibit (i) non-stationary regime changes, (ii) coexisting low- and high-frequency components with

[1] "Configurable" indicates that LR–HR pairs can be constructed by running the simulator at different sampling settings and/or applying controlled downsampling, but a standardized paired SR benchmark (multi-factor LR versions aligned to a fixed HR target) is not typically distributed as part of the resource.

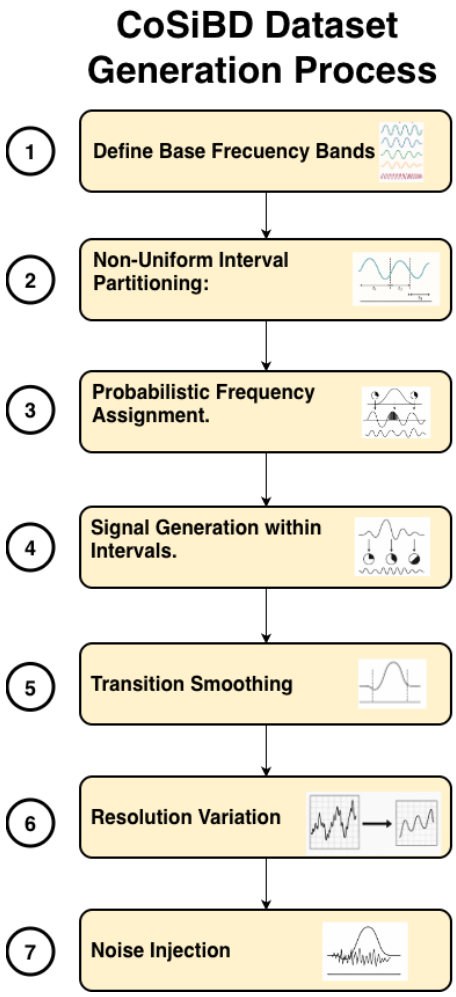


Figure 1. Schematic overview of the CoSiBD signal generation process.

Resource	Domain	Form	Paired LR–HR SR	Multi-resolution	Noise / artifacts	Reproducibility granularity
CoSiBD (this work)	Generic time series (complex-structured signals)	Dataset + generator	Yes (LR → HR targets)	Yes (150/250/500/1000 → 5000)	Gaussian + structured interference; primary benchmark uses direct decimation	Per-signal metadata; deterministic regeneration (seed-controlled)
RadioML 2016.10A[18,19]	Wireless communications (I/Q)	Dataset	No (classification benchmark)	N/A (not SR)	Variable SNR + channel impairments	Dataset-level (labels/SNR); not per-sample "recipe"
RadioML 2018.01A[20]	Wireless communications (I/Q)	Dataset	No (classification benchmark)	N/A (not SR)	Simulated channel effects + SNR variability	Dataset-level; not SR-paired
ECGSYN[21,22]	ECG (physiology)	Simulator/tool	Configurable[1]	Configurable (via sampling settings)	Model-based; supports controlled variability	Configurable via simulator parameters (user-defined)
SEREEGA[23]	EEG (physiology)	Simulator/ toolbox	Configurable[1]	Configurable (user-defined)	Supports noise and event-related components	Configurable via simulator parameters (user-defined)
LoadGAN[24]	Power systems load time series	Generator/tool	No (generation)	Yes (variable sampling rates)	Domain-specific variability (load patterns)	Tool-based; generation is configurable
Synthetic LR–HR seismic traces (example)[25]	Seismic traces (geophysics)	Paper-specific paired data	Yes (LR–HR pairs)	Typically limited (study-specific)	Study-dependent	Paired data available for the study; limited generality

Table 1. Representative publicly available synthetic time-series datasets and simulators related to signal processing and learning. "Form" indicates whether the resource is distributed primarily as a fixed dataset or as a simulator/generator. "Reproducibility granularity" summarizes whether exact per-sample regeneration is supported via documented parameters and seeds.

intermittent transients, (iii) smooth amplitude-envelope evolution, and (iv) slow baseline drift and measurement noise. CoSiBD instantiates these properties via non-uniform interval partitioning with change-points, separate low/high-frequency bands, spline-based envelopes and frequency profiles, and explicit offset/noise terms. Figure 2 provides qualitative examples of these motivating properties; the goal is to capture challenging structure for SR benchmarking rather than match a specific domain distribution.

The signal generation pipeline involves the following steps:

1. **Base frequency band definition:** A set of distinct frequency bands is defined to represent the underlying spectral content of the signals. These can be adjusted to reflect application-specific characteristics.

2. **Non-uniform interval partitioning:** The total signal duration is divided into multiple intervals of variable length. The interval lengths are determined probabilistically to introduce variability in the signal structure.

3. **Frequency assignment:** Each interval is assigned a dominant frequency band, sampled according to a predefined probability distribution. This introduces spectral variation over time.

4. **Signal synthesis:** A sinusoidal waveform, or a combination of sinusoids within the assigned frequency band, is generated for each interval. Signal parameters such as amplitude and phase are configurable.

5. **Transition smoothing:** To avoid discontinuities at interval boundaries, a smoothing function is applied to overlapping segments. This ensures gradual transitions between intervals with different frequency content.

6. **Resolution variation:** All signals are initially synthesized at a high temporal resolution (5,000 samples over the domain $[0, 4\pi]$). Lower-resolution versions are created using simple decimation (uniform subsampling). This keeps the SR task aligned with reconstructing the original high-resolution target; the low-resolution observation is obtained by subsampling

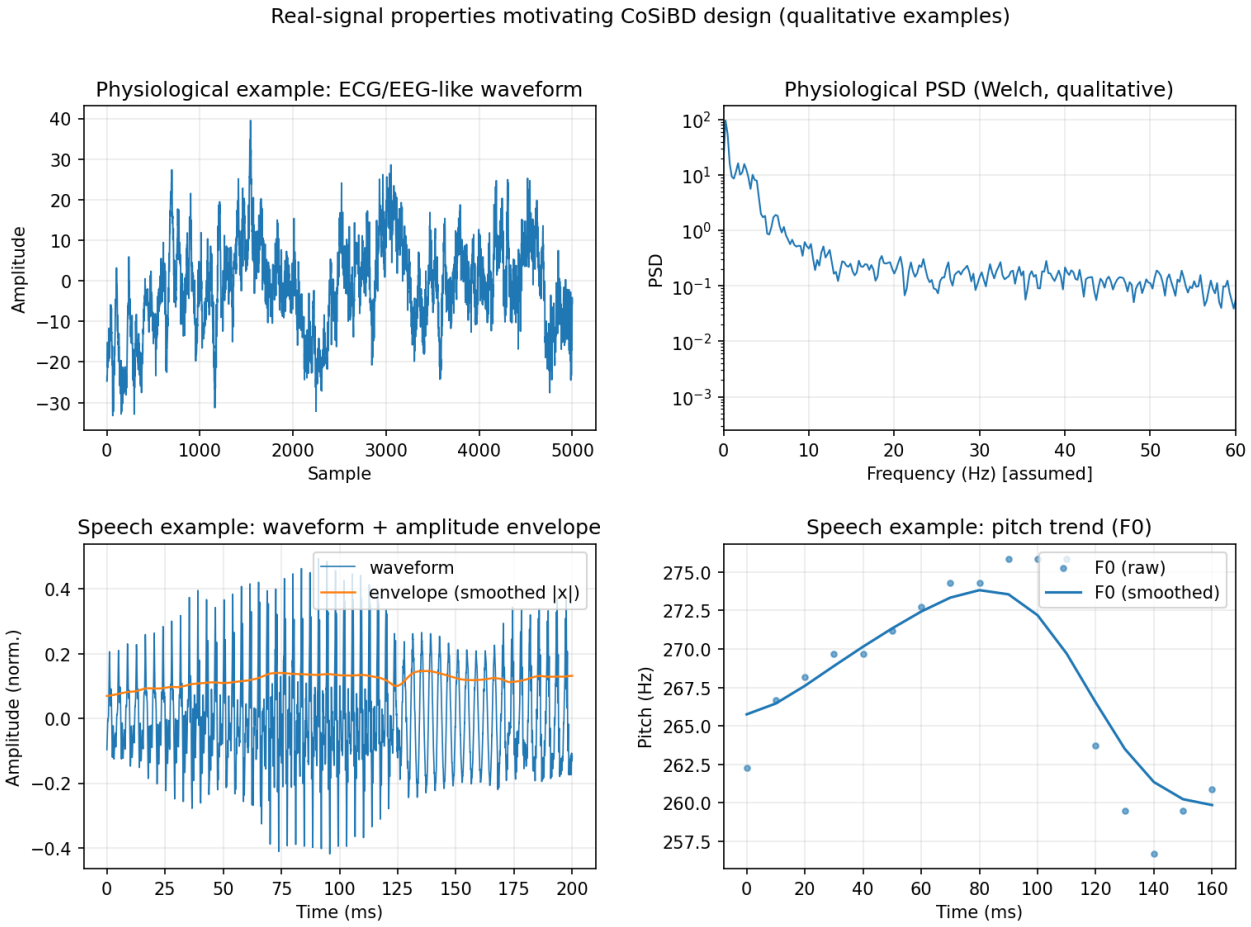


Figure 2. Qualitative real-signal properties motivating the CoSiBD design. The physiological example illustrates non-stationarity in the waveform and structured spectral content; the speech example illustrates amplitude-envelope dynamics and a smoothly varying pitch (F0) trend. These observations motivate CoSiBD mechanisms such as regime partitioning with change-points, low/high-frequency bands, and spline-based envelopes/frequency profiles.

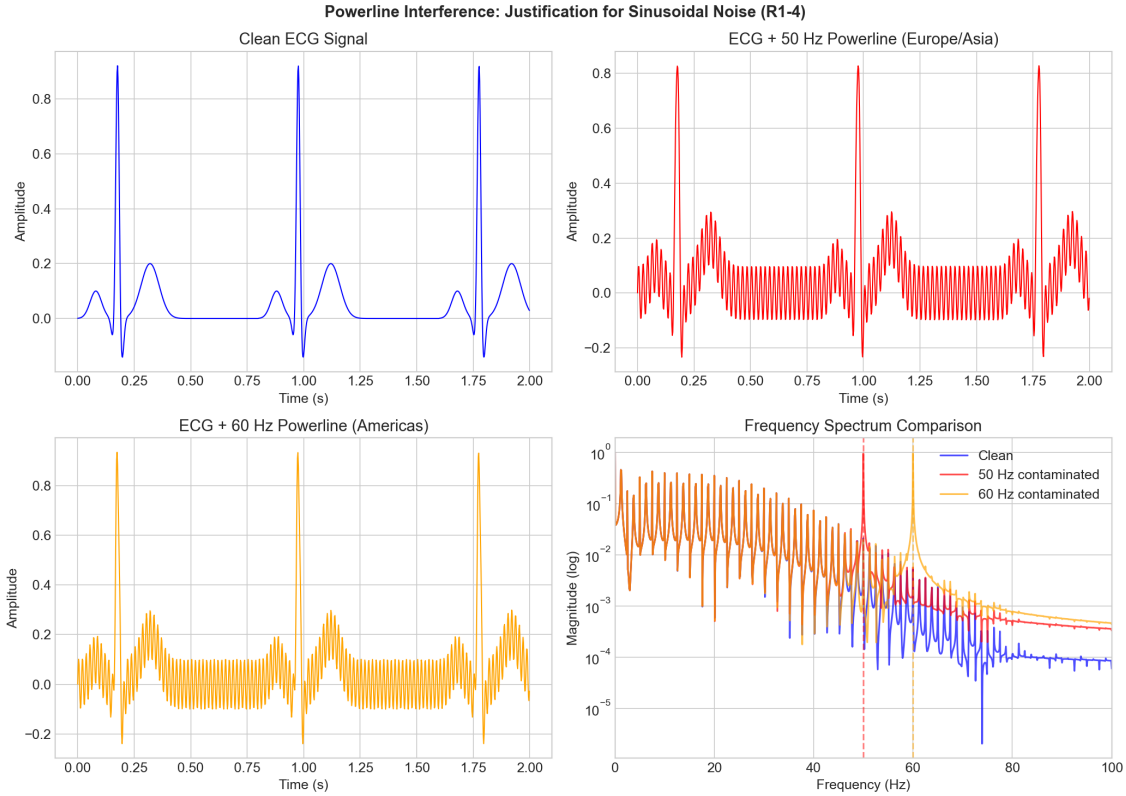


Figure 3. Qualitative motivation for the structured interference term used in CoSiBD. An illustrative example shows how adding a narrow-band sinusoidal component (interpretable as 50/60 Hz under the illustrative convention $T = 4\pi$ s) produces the characteristic periodic contamination observed in real recordings, while broadband noise captures the measurement floor.

the original sequence without pre-filtering. Reconstructing low-pass filtered signals is not an objective of CoSiBD. For reproducibility, given a high-resolution sequence $x_{\mathrm{HR}}[n]$ of length $N = 5000$ and a target low-resolution length $M \in \{1000, 500, 250, 150\}$, we form $x_{\mathrm{LR}}[i] = x_{\mathrm{HR}}[n_i]$ using the fixed index set $n_i = \left\lfloor \frac{i(N-1)}{M-1} + 0.5 \right\rfloor$ for $i = 0, \ldots, M-1$ (applied identically to the time array). This reduces to standard stride decimation when M divides N.

7. **Noise injection:** Controlled levels of synthetic noise are added to the signals to emulate different data acquisition scenarios. Two noise types are implemented: Gaussian noise with configurable standard deviation (relative to signal amplitude) and structured sinusoidal noise bursts (deterministic sinusoidal components). Noise is applied probabilistically with 50% probability per signal. Both the type and intensity of the noise can be configured.

Rationale for structured 50/60 Hz interference and noise. Real measurement pipelines frequently contain narrow-band interference (e.g., mains hum) superimposed on broadband sensor noise. To reflect this common acquisition artifact, CoSiBD includes an optional structured sinusoidal component in addition to Gaussian noise. CoSiBD signals are generated over a reference domain (by default $\tau \in [0, 4\pi]$); interpreting τ as physical time (and therefore reporting frequencies in Hz) requires an explicit time scaling. Throughout this manuscript we adopt an illustrative convention that maps the reference domain to a duration $T = 4\pi$ seconds, under which the structured component can be interpreted as a 50/60 Hz-like powerline interference term, while the broadband term represents the measurement noise floor. Figure 3 illustrates this qualitative motivation; the intent is not to reproduce a specific device transfer function but to include realistic nuisance factors that SR models must handle.
Sampling units and frequency interpretation. CoSiBD signals are provided as discrete sequences $x[n]$ (e.g., $N = 5{,}000$ samples) that are directly used as inputs/targets by SR models. The internal generation domain $\tau \in [0, 4\pi]$ is a reference parameterization; interpreting it as physical time requires choosing a duration T (in seconds) for the reference interval. Under this convention, the implied sampling rate is $f_s = N/T$ and all frequencies reported in Hz scale linearly with $4\pi/T$. Throughout this manuscript, when reporting example frequencies in Hz we adopt the illustrative convention $T = 4\pi$ s, yielding $f_s \approx 5000/(4\pi) \approx 398$ Hz; other equally valid mappings exist depending on application. Consequently, any band-specific interpretation in Hz (e.g., "low/high" frequency ranges) should be understood under the chosen T; changing T rescales all

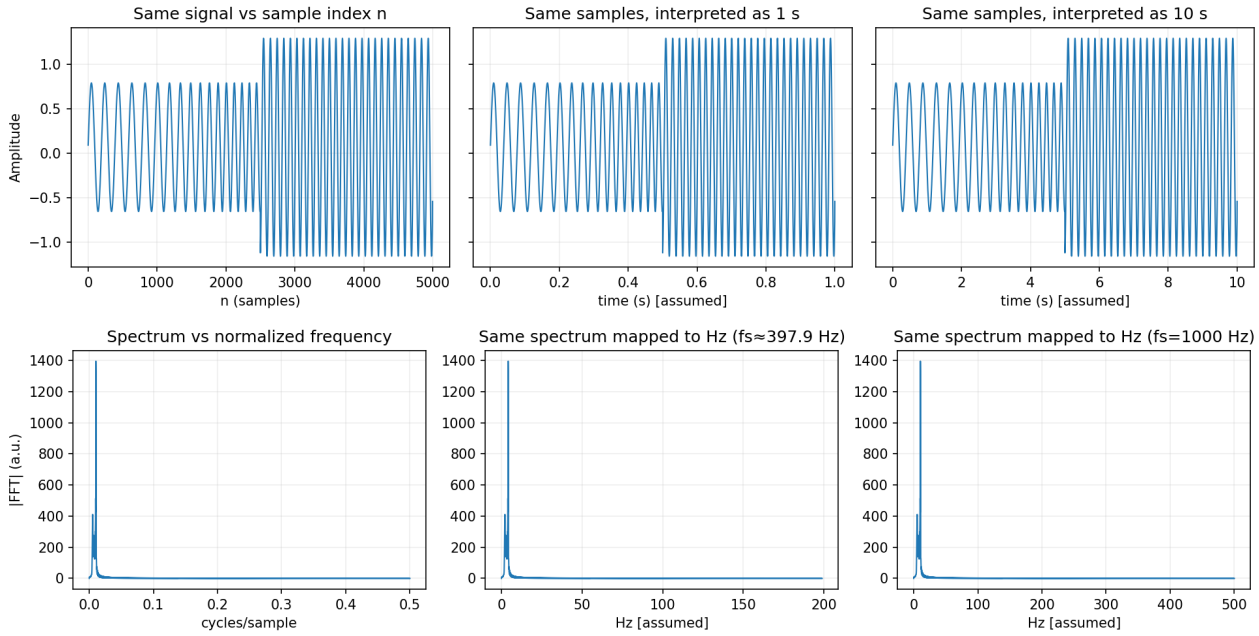


Figure 4. Sampling/unit convention in CoSiBD. Top: the same discrete sequence $x[n]$ can be plotted against the sample index or under different assumed time scalings. Bottom: the intrinsic frequency axis is normalized (cycles/sample); mapping to "Hz" depends on the assumed sampling rate f_s (two example mappings shown).

reported Hz values while preserving the underlying discrete sequences, which is a key feature of CoSiBD's reference-domain design. Figure 4 illustrates that the discrete samples are unchanged under different time scalings and that Hz axes shift with the assumed f_s, while the normalized spectrum (cycles/sample) is invariant.

The parameters that govern each step of the generation process—such as interval length distributions, frequency band selection probabilities, smoothing function characteristics, sampling rates, and noise settings—can be configured to produce signal sets tailored to different domains or experimental conditions. All generation parameters, including random seeds, are documented in comprehensive metadata (`signals_metadata.json`), enabling exact reproduction of individual signals or the complete dataset. The generation pipeline is implemented in modular Python code available in the SignalBuilderC package, with clear interfaces for customization and extension. These configurations are included in the dataset's accompanying code to support reproducibility and allow users to regenerate the signals under consistent conditions.

Data Records

The Complex Signal Benchmark Dataset (CoSiBD) is publicly available on Zenodo[26] and consists of synthetic temporal signals created to support the development and evaluation of temporal super-resolution (SR) algorithms. This section provides an overview of the dataset structure, content, and storage format.

The dataset comprises 2,500 high-resolution signals, each with corresponding subsampled versions at four resolution levels, organized into three main categories:

- **High-resolution signals**: 2,500 signals with 5,000 samples each, spanning the domain $[0, 4\pi]$. Each signal is stored in three formats: NumPy compressed format (.npz), plain text (.txt), and JSON (.json). Per-signal metadata (frequency profiles with explicit change-points (`base_points` and `high_freq_points`) and segment labels (`variation_type`), amplitude envelopes, spline parameters, vertical offsets, noise configurations, and random seeds) is provided in a consolidated JSON file (`signals_metadata.json`) with one entry per signal, enabling exact regeneration.

- **Simple subsampled signals**: Uniform decimation (uniform subsampling) of each signal to four target resolutions (150, 250, 500, and 1,000 samples). These low-resolution versions serve as inputs for SR benchmarking against the original

5,000-sample target. Stored in .npz, .txt, and .json formats.

Reproducibility is ensured through documented random seeds: each high-resolution signal is generated using a unique seed (ranging from 10,000 to 12,499), enabling exact regeneration of individual signals or the entire dataset. All generation parameters are stored in metadata JSON files, including: (1) frequency profile parameters—tau_frequency values from uniform distribution [1, 2] with 0.05 step; (2) amplitude envelope parameters—tau_amplitude from {1, 3, 5, 8, 10, 12, 15, 20} for tension splines, or zero-order step functions (70% probability); (3) vertical offsets—normally distributed (mean=0, SD=3.0); and (4) noise configurations—50% probability of Gaussian or structured noise.

The dataset is provided as consolidated files under `SignalBuilderC/data/`. High-resolution signals are stored as `signals_high_resolution_5000.[npz|txt|json]`. Simple subsampled (decimated) signals are stored as `signals_subsampled_simple_{150,250,500,1000}.[npz|txt|json]`. Dataset-level metadata and configuration are stored in `signals_metadata.json`, `signals_metadata_consolidated_2500.json`, and `dataset_summary.json`.

Each signal is stored in three formats: (1) NumPy compressed format (.npz) containing the signal array, time array, and (for high-resolution only) clean signal without noise; (2) consolidated plain text format (.txt) with one signal per row (samples separated by whitespace) for maximum portability; and (3) JSON format (.json) with both time and signal arrays for web-based applications and interoperability. Per-signal metadata is provided in `signals_metadata.json` (one entry per signal), and dataset-level configuration is provided in `dataset_summary.json`.

Metadata schema and example

CoSiBD provides per-signal metadata to support (i) deterministic regeneration, (ii) principled partitioning (e.g., by noise type/level or segment labels), and (iii) analysis of the piecewise structure induced by change-points. Table 2 summarizes representative fields contained in `signals_metadata.json`. A minimal example entry is shown below (one signal; values truncated for brevity).

Field	Type / example	Meaning
`signal_id`, `index`	`"signal_0000"`, 0	Unique identifier and row index used to align LR–HR pairs across consolidated files.
`seed`	10000–12499	Per-signal random seed enabling deterministic regeneration.
`t_start`, `t_end`	0.0, 12.566...	Reference-domain interval ($\tau \in [0, 4\pi]$) used by the generator.
`fs_high`	397.887...	Illustrative sampling rate under the manuscript convention $T = 4\pi$ s (see Sampling units and frequency interpretation).
`tau_frequency`	1.15	Frequency-profile spline tension parameter.
`amplitude_spline_type`, `tau_amplitude`	`"zero_order"`, `"N/A"`	Envelope model type and, when applicable, its tension parameter.
`base_points`	$K \times 2$ array	Change-points and base-band frequencies defining the piecewise frequency profile.
`high_freq_points`	$K \times 2$ array	Change-points and high-frequency component levels (intermittent transients).
`variation_type`	`["low", ...]`	Segment labels aligned to change-points (e.g., low/high/no-change regime).
`amp_knots`, `amp_values`	arrays	Envelope control knots and values.
`vertical_offset`	0.069...	Additive baseline drift term.
`noise_profile`	JSON object	Noise flags and parameters (e.g., Gaussian vs structured interference; probabilities; amplitudes).

Table 2. Representative per-signal metadata fields in `signals_metadata.json`. The file contains one entry per signal, supporting deterministic regeneration and analysis/partitioning based on the signal's piecewise structure and nuisance settings.

{

```
    "t_start": 0.0,
    "t_end": 12.566370614359172,
    "fs_high": 397.88735772973837,
    "tau_frequency": 1.15,
    "amplitude_spline_type": "zero_order",
    "vertical_offset": 0.06905161748158965,
    "base_points": [[0.0, 2.076409156965817], [1.9229451245119575, 2.076409156965817], ...],
    "high_freq_points": [[0.0, 0.0], [1.9229451245119575, 0.0], ...],
    "variation_type": ["low", "low", "low", "low"],
    "amp_knots": [0.0, 6.28192867011815, 12.5638573402363],
    "amp_values": [0.7237770021649202, 1.2266792057266251, 0.9661294815645534],
    "noise_profile": {"has_noise": true, "noise_type": "gaussian", "p_has_noise": 0.5, ...},
    "seed": 10000,
    "signal_id": "signal_0000",
    "index": 0
}
```


The following resolution levels are available:

- **High-resolution:** 5000 samples per signal, sampled over the reference domain $\tau \in [0, 4\pi]$. Under the illustrative convention $T = 4\pi$ s (seconds), this corresponds to $f_s = 5000/(4\pi) \approx 398$ Hz.

- **Subsampled resolutions:** Available as simple decimated versions:

  - 1000 samples (illustrative $f_s \approx 79.6$ Hz for $T = 4\pi$ s)

  - 500 samples (illustrative $f_s \approx 39.8$ Hz for $T = 4\pi$ s)

  - 250 samples (illustrative $f_s \approx 19.9$ Hz for $T = 4\pi$ s)

  - 150 samples (illustrative $f_s \approx 11.9$ Hz for $T = 4\pi$ s)

Table 3 outlines the main parameters used in signal generation. Each high-resolution signal was generated with a unique random seed (10,000–12,499) and randomly sampled parameter values within the defined ranges, supporting diversity while maintaining reproducibility.

Parameter	Range	Description
Low Frequency	1–5 (illustrative Hz for $T = 4\pi$ s)	Low-frequency component present in signals
High Frequency	20–100 (illustrative Hz for $T = 4\pi$ s)	Higher-frequency variations for transitions
Change Points	2–11	Number of frequency transitions per signal
Change Locations	Random	Time locations where transitions occur
Variation Type	Categorical	Defines nature of frequency change ("low", "high", "no_change")
Amplitude Range	3–16	Range for amplitude envelope values
Vertical Offset	N(0, 3.0)	Normally distributed offset added to signals
Spline Type	Mixed	70% zero-order (step), 30% tension spline
Tension Parameter (freq)	[1, 2]	Tau values for frequency spline interpolation
Tension Parameter (amp)	{1,3,5,8,10,12,15,20}	Tau values for amplitude spline (when tension type)
Noise Probability	50%	Probability of adding noise to each signal
Random Seed	10000–12499	Unique seed per signal for reproducibility

Table 3. Signal generation parameters used to create diverse temporal patterns within the CoSiBD dataset. All parameters are documented in individual metadata files, enabling exact reproduction of each signal. These parameters control the frequency composition and temporal structure.

To explicitly characterize dataset diversity and complexity, CoSiBD spans multiple controlled axes of variation (Table 3), including the number and location of change points, categorical transition types, low/high frequency bands, and amplitude-envelope configurations. The resulting variability is visible in representative realizations (Figures 5 and 6) and is quantified in Technical Validation via the distribution of dominant frequencies (Figure 7 and Table 4) and PSD behavior under different resolutions and noise settings (Figures 9 and 10). While the dataset is synthetic and not fitted to match a single domain-specific distribution, these controlled variations provide reproducible coverage of common real-world time-series phenomena such as non-stationarity, transient high-frequency events, and additive noise.
Figure 5 shows a representative signal from the dataset sampled at different resolution levels, as well as a version with added noise. This illustrates the variety of sampling and noise conditions included in CoSiBD.

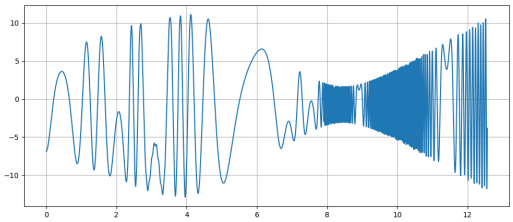
Figure 6 displays four additional synthetic signals generated using different configuration parameters. These examples demonstrate the variability in temporal structure across instances in the dataset.
The full dataset is hosted in Zenodo[26] (DOI: 10.5281/zenodo.15138853) and includes the signal files and associated metadata in structured folders.

Technical Validation

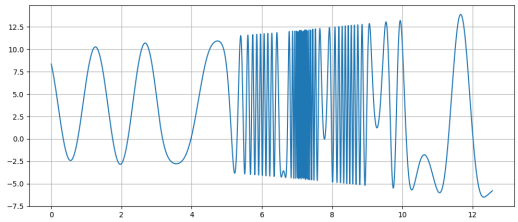
This section evaluates the signal generation procedure by analyzing spectral properties under different conditions, including the distribution of dominant frequencies, spectral stability across sampling rates, and the effect of noise. These analyses aim to assess variability and stability under the reported settings, and to document the dataset's behavior for reproducible use. Below, the methodologies and results are described in detail.

Validation Context

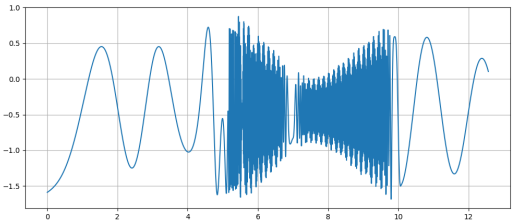
Experimental parameters were selected to support reproducibility and to illustrate representative behaviors of the generator under the reported settings. The number of signals (n=50) provides a compact but informative sample to summarize variability in spectral characteristics. Sampling resolutions (150, 250, 500, and 1000 samples) reflect scenarios requiring different levels of detail, aligning with typical signal processing use cases. Noise amplitudes and other parameter ranges were motivated by common acquisition artifacts and exploratory checks, with the goal of providing a controllable benchmark rather than an exhaustive model of any specific measurement pipeline.



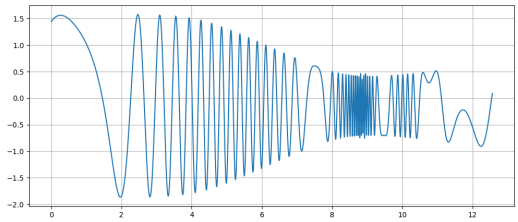
(a) High-resolution signal (5000 samples).



(b) Medium-resolution signal (500 samples).

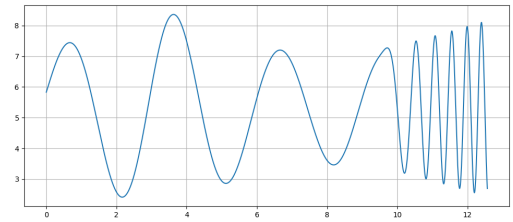


(c) Low-resolution signal (250 samples).

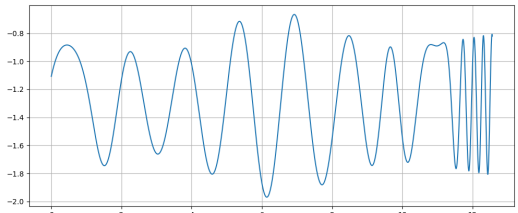


(d) Signal with added noise.

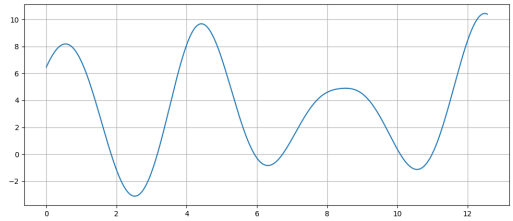
Figure 5. A synthetic signal sampled at different resolutions: (a) high (5000 samples), (b) medium (500 samples), (c) low (250 samples), and (d) with added noise. These examples reflect the multi-resolution and noise conditions present in the dataset.



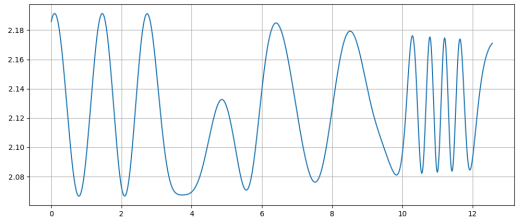
(a) Signal with increasing frequency over time.



(b) Signal with localized frequency variation.



(c) Signal with smooth oscillations and broad amplitude cycles.



(d) Signal with irregular peak spacing.

Figure 6. Examples of synthetic signals in the dataset generated with different parameter configurations. Each signal presents a distinct temporal profile.

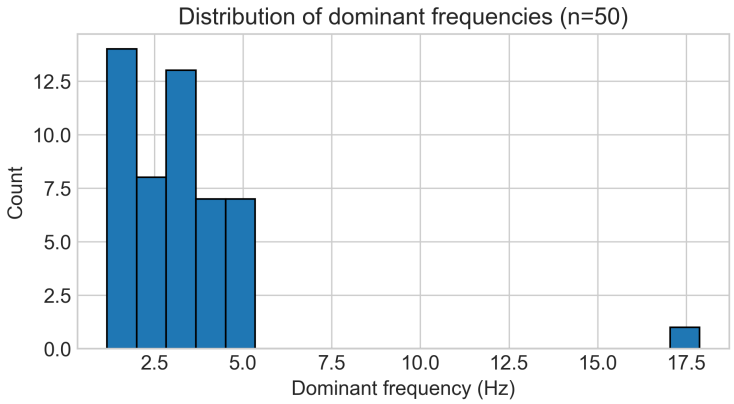


Figure 7. Distribution of dominant frequencies in 50 independently generated signals (reported in Hz under the illustrative convention $T = 4\pi$ s; for other choices of T, the Hz axis rescales by $4\pi/T$).

Statistic	Value (Hz; illustrative $T = 4\pi$ s)
Average Dominant Frequency	0.508
Standard Deviation	0.195
Minimum Dominant Frequency	0.390
Maximum Dominant Frequency	1.171

Table 4. Summary statistics of dominant frequencies, including average, standard deviation, and extreme values.

Analysis of Dominant Frequency Distribution

To assess the stability and variability of the primary spectral components, we analyzed the distribution of dominant frequencies across multiple generated signals. A total of fifty independent signals were synthesized using identical input parameters. To examine their spectral characteristics, we computed the power spectral density (PSD) of each signal, which quantifies how signal power is distributed across different frequencies.

The PSD was estimated using Welch's method, selected for its ability to reduce noise and provide a smoother spectral representation[27]. This method stabilizes spectral estimation by dividing the signal into overlapping segments, computing their individual spectra, and averaging them. This reduces variance from random fluctuations and yields a smoother estimate. For each signal, the dominant frequency was identified as the frequency at which the PSD reaches its maximum value. This corresponds to the most prominent spectral component, indicating where the signal concentrates most of its energy.

By analyzing the distribution of dominant frequencies across the dataset, we evaluate whether the generated signals exhibit consistent spectral patterns or if there is significant variation. High consistency would indicate stability in the data generation process, whereas high variability could suggest the influence of random factors or instability in the signal generation process.

The results, shown in Figure 7 and Table 4, show that the dominant frequency values (reported in Hz under the illustrative convention $T = 4\pi$ s) are concentrated in a low-frequency range, with occasional higher-frequency occurrences under the same convention. For other choices of T, these values rescale linearly by $4\pi/T$. This behavior reflects the method's ability to generate signals with consistent primary structures while introducing controlled variability.

Figure 8 presents examples of signals from the CoSiBD dataset with increasing levels of added noise, illustrating how amplitude fluctuations progressively obscure the underlying temporal structure.

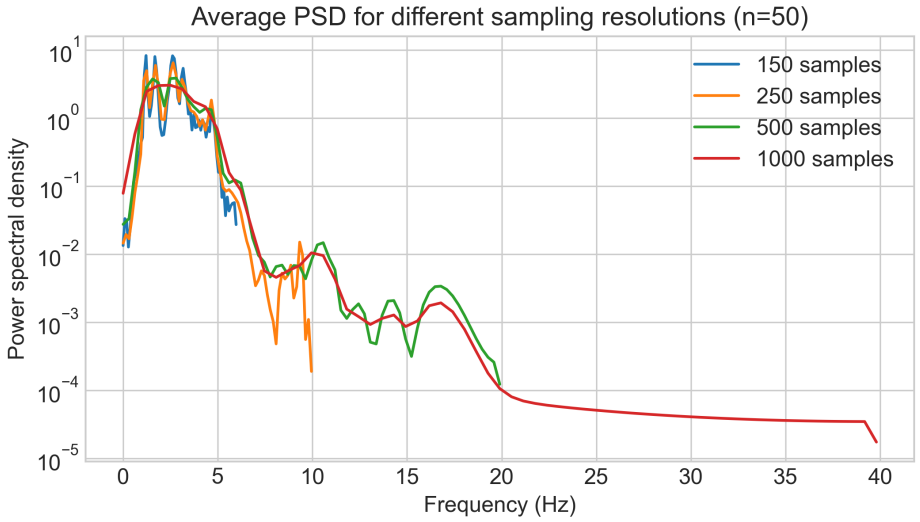
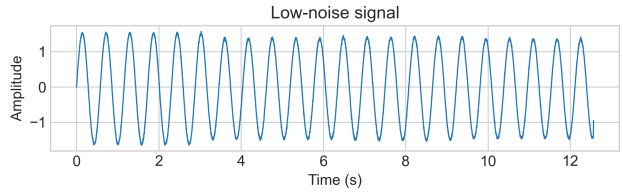
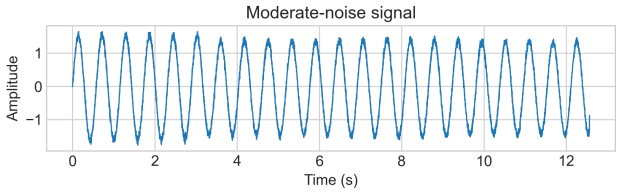


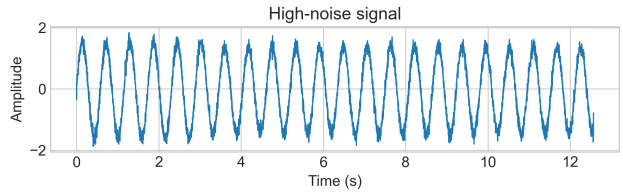
Figure 9. Average power spectral density (PSD) for different sampling resolutions based on 50 independent runs (Hz axis under the illustrative convention $T = 4\pi$ s).



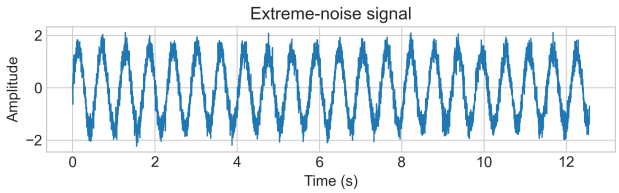
(a) Low-noise signal, where amplitude variations are present but minimally distorted.



(b) Moderate-noise signal, with irregular peaks and troughs beginning to distort the oscillatory pattern.



(c) High-noise signal, where significant distortion leads to unpredictable fluctuations.



(d) Extreme-noise signal, where the original oscillatory structure is almost entirely masked by chaotic interference.

Figure 8. Visualization of signals under increasing noise conditions, showing how added noise progressively masks the original temporal patterns. From low (a) to extreme noise levels (d), this degradation highlights reconstruction challenges for super-resolution models.

Spectral Stability Across Sampling Resolutions

This analysis aims to investigate the influence of sampling resolution (number of samples) on the robustness of spectral estimates under varying frequency content. When frequency axes are reported in Hz, they follow the illustrative convention $T = 4\pi$ s; for other choices of T, the Hz axis rescales by $4\pi/T$. At lower resolutions, reduced sampling density and coarser frequency grids can obscure or merge spectral peaks, compromising the ability to distinguish closely spaced spectral components[28]. Conversely, higher resolutions improve the granularity of the frequency axis, allowing for better separation of spectral features and reducing the risk of misrepresenting the signal's underlying structure[29].

This evaluation documents how spectral summaries vary with sampling resolution under the reported settings. The intent is to provide descriptive context for using CoSiBD at different resolutions (and computational budgets) in benchmark protocols, rather than to prescribe a universal sampling rate.

As shown in Figure 9, lower sampling resolutions, specifically the blue curve (150 samples) and the orange curve (250

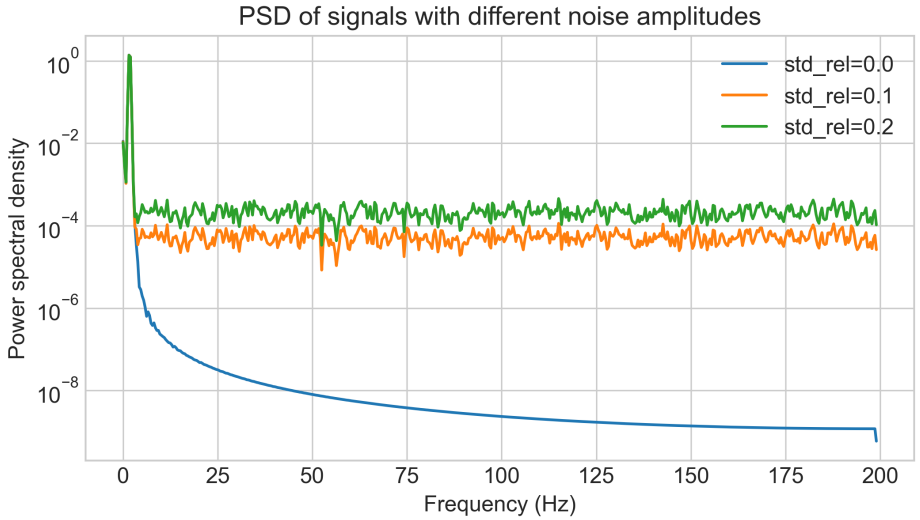


Figure 10. Power spectral density (PSD) of signals generated with different noise amplitudes (Hz axis under the illustrative convention $T = 4\pi$ s).

samples), exhibit a noticeable reduction in detail within the higher-frequency range (reported in Hz under the illustrative convention $T = 4\pi$ s). These lower-resolution curves display greater fluctuations, particularly at higher frequencies under this convention, which is consistent with the theoretical effects of subsampling. The blue curve (150 samples) is especially affected, showing significant variability and a less stable spectral representation in the higher frequencies.

In contrast, the higher sampling resolutions demonstrate a smoother and more stable spectral profile across all frequencies. The red curve (1000 samples), in particular, captures finer details and exhibits minimal high-frequency noise, making it the smoothest estimate among the reported settings.

Impact of Noise on Frequency Characteristics

We analyze how varying the noise amplitude affects the power spectral density (PSD), with particular attention to differences between low- and high-frequency regions.

Figure 10 illustrates the impact of different noise amplitudes on the Power Spectral Density (PSD) under the reported settings (Hz axis under the illustrative convention $T = 4\pi$ s). As the noise amplitude increases—from 0.0 (blue curve) to 0.2 (red curve)—the estimated PSD exhibits increased variability at higher frequencies, while the low-frequency region remains comparatively stable in these plots.

Across these settings, the low-frequency region changes less than the higher-frequency region in these estimates. This observation provides context for the subsequent super-resolution benchmark, where both time-domain and frequency-domain metrics are reported.

Multi-Scale Super-Resolution Benchmark

To illustrate a baseline use case of CoSiBD and provide reference results across a range of upsampling factors, we trained a series of convolutional neural network (CNN) models for time series super-resolution at four different scaling factors: 5×, 10×, 20×, and 33×. All models employed the TimeSeriesSRNet architecture—a five-layer encoder-decoder network with 1D convolutional layers (kernel size 5, ReLU activations) and bilinear upsampling. For this benchmark, the 2,500 high-resolution signals were partitioned into an experiment-specific split of 2,000 paired signals for training (low-resolution input to 5,000-sample high-resolution target) and 500 held-out signals for validation. This split is used only for the reported protocol and is not distributed as a predefined dataset partition. Each model was trained using mean squared error (MSE) loss, Adam optimizer (learning rate 0.001, weight decay 10^{-5}), batch size 16, and early stopping with patience of 3 validation checks (every 10 epochs). Training was conducted on Apple Silicon GPU (MPS backend) to accelerate convergence.

Table 5 summarizes the validation performance, convergence characteristics, and computational requirements for each upsampling factor. In these runs, all models completed the 50-epoch budget and showed stable validation loss trends, with the lowest-resolution inputs (150 samples, 33× upsampling) requiring the most epochs to achieve stable performance. Validation

loss increased systematically with upsampling factor, reflecting the inherent difficulty of reconstructing fine temporal details from severely undersampled inputs (Table 5, Figure 11).

Input Size	Factor	Val Loss	Epochs	Early Stop	LSD	SCORR
1000 samples	5×	0.0845	50	No	0.51±0.63	0.98±0.10
500 samples	10×	0.1524	50	No	0.64±0.63	0.98±0.10
250 samples	20×	0.4376	50	No	0.95±0.67	0.98±0.10
150 samples	33×	1.0326	50	No	1.21±0.67	0.98±0.11

Table 5. Multi-scale super-resolution benchmark results. Validation loss measured as mean squared error on 500 independent validation signals. LSD (Log Spectral Distance) quantifies spectral content deviation (lower is better), while SCORR (Spectral Correlation) measures frequency-domain similarity (higher is better, range [0,1]). Early Stop indicates whether training terminated before maximum epochs. All models completed the full 50-epoch training without early termination, showing stable convergence across all upsampling factors.

To complement amplitude-based validation with frequency-domain assessment, we computed spectral fidelity metrics for all reconstructed signals. Log Spectral Distance (LSD) increased from 0.51 (5×) to 1.21 (33×), while Spectral Correlation (SCORR) remained consistently high (Table 5, Figure 14). Figure 13 presents representative spectrogram comparisons across all upsampling factors, illustrating how reconstruction artifacts become more visible at higher upsampling factors.

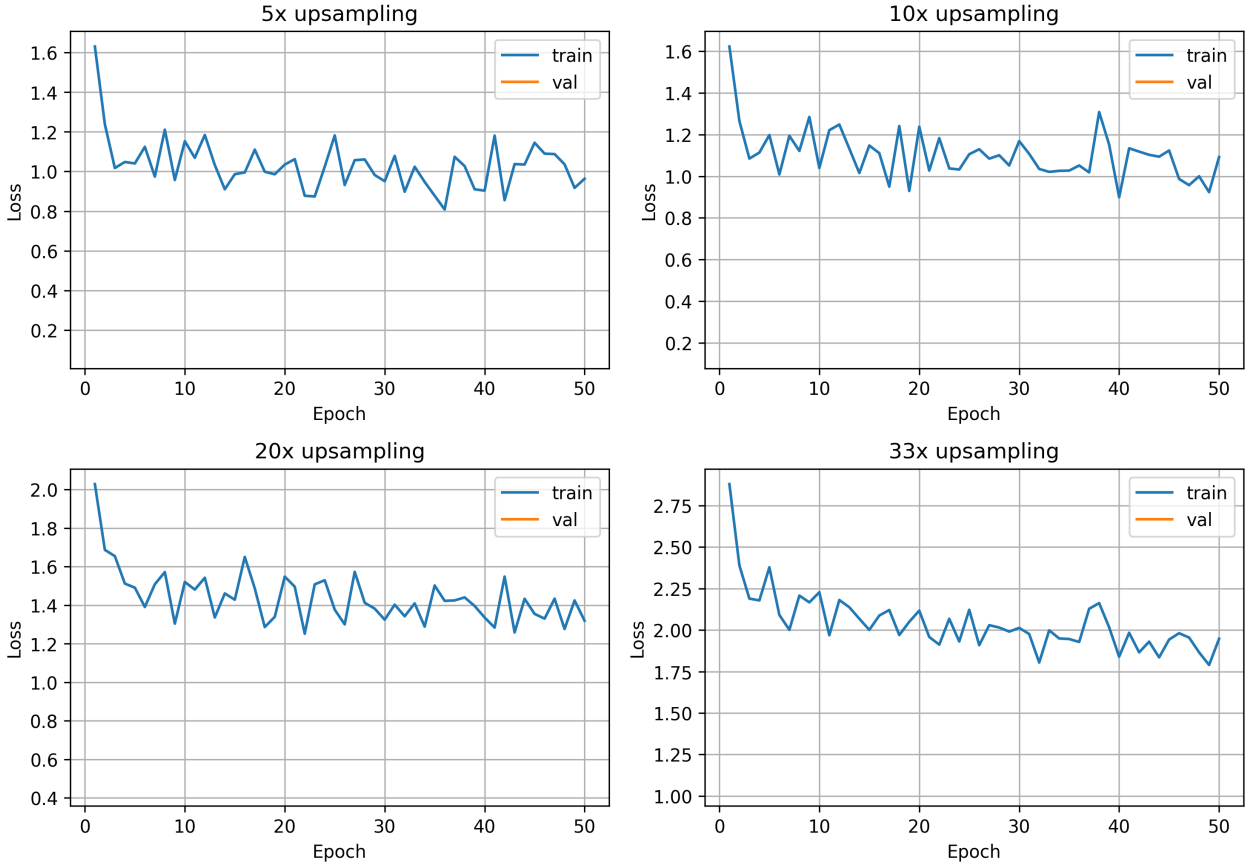


Figure 11. Training and validation loss evolution across all four upsampling factors (5×, 10×, 20×, 33×). Each panel shows loss curves during training; in these runs, training and validation curves follow similar trends without pronounced divergence. The systematic increase in final validation loss with upsampling factor reflects the inherent difficulty of reconstructing fine temporal details from severely undersampled inputs.

Figure 11 illustrates the training and validation loss evolution for all four upsampling factors. Representative prediction

examples (Figure 12) provide qualitative comparisons of reconstructed outputs against ground truth across scaling factors.

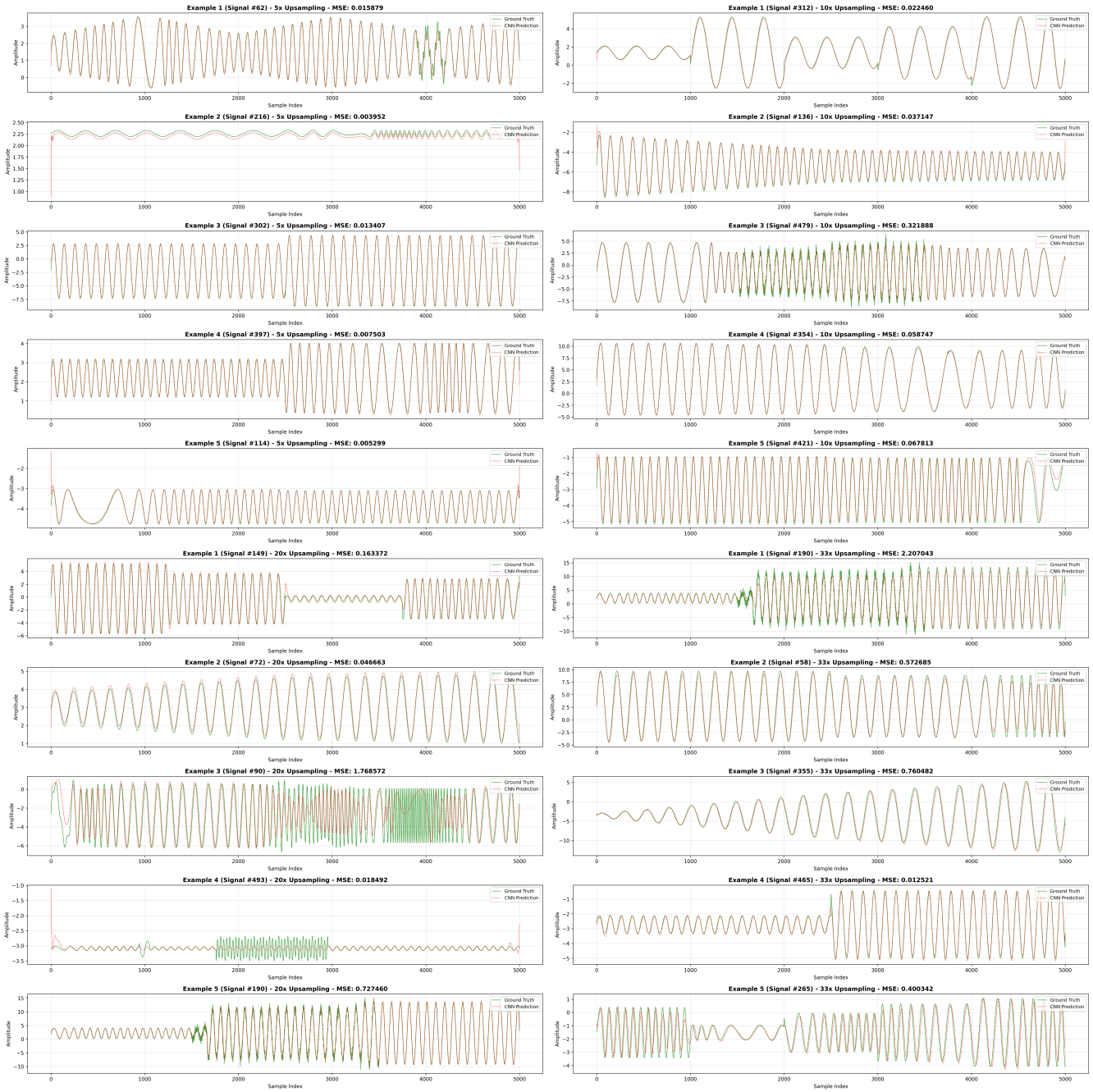


Figure 12. Representative prediction examples across all upsampling factors. Each quadrant shows prediction comparisons for a different scaling factor (5×, 10×, 20×, 33×), displaying low-resolution inputs, ground-truth high-resolution signals, and CNN-reconstructed outputs.

These multi-scale experiments provide quantitative baseline results for future benchmarking studies. The systematic increase in task difficulty—from moderate 5× upsampling to extreme 33× reconstruction—provides a reference protocol for comparing architectures, loss functions, and training strategies in the time series super-resolution domain.

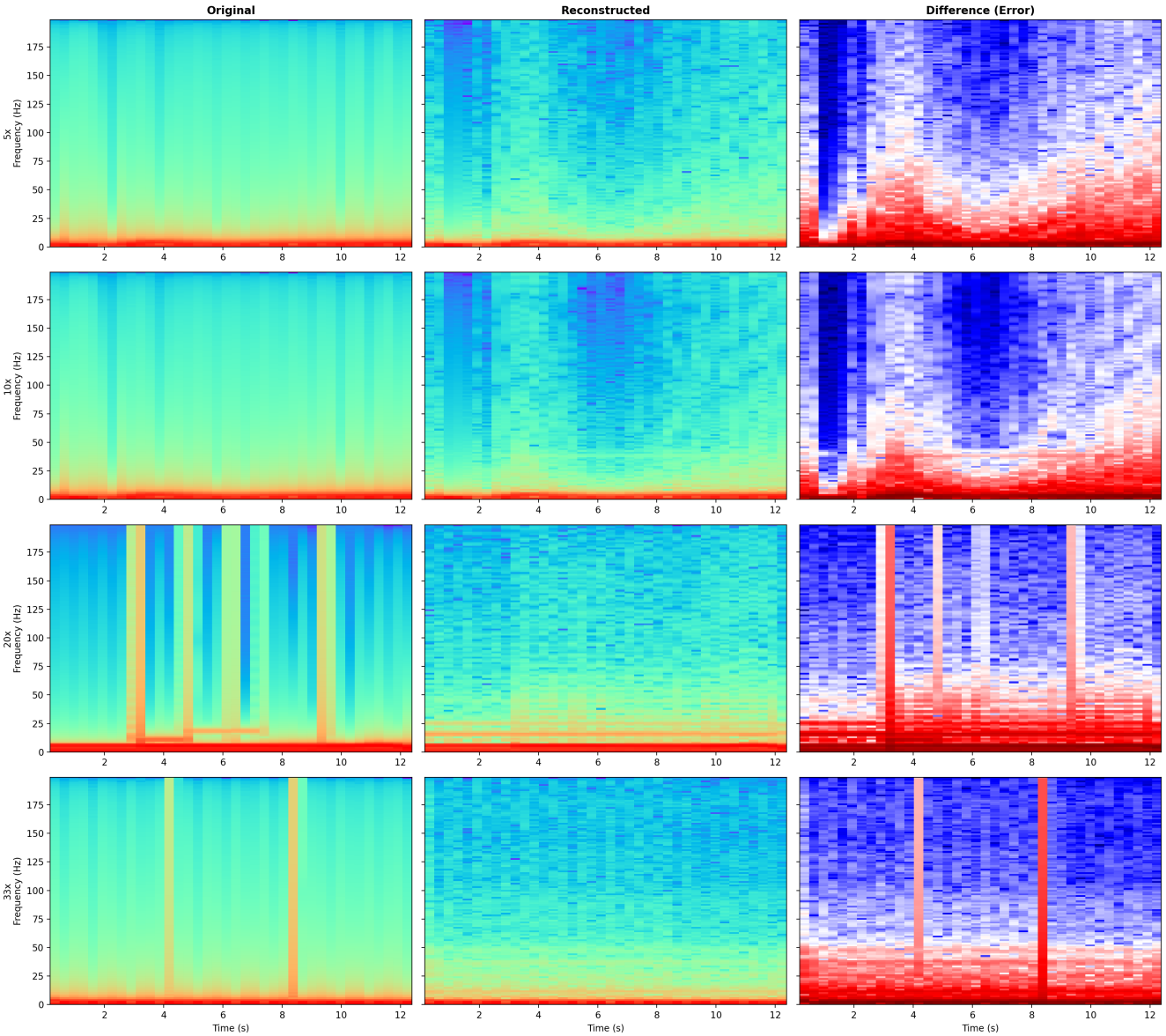


Figure 13. Spectrogram comparison across all upsampling factors. Each row represents a different upsampling factor (5×, 10×, 20×, 33×), showing original signal (left), CNN-reconstructed signal (center), and spectral difference (right). Reconstruction artifacts become more visible at higher upsampling rates. Representative signals selected based on median Log Spectral Distance (LSD) for each factor.

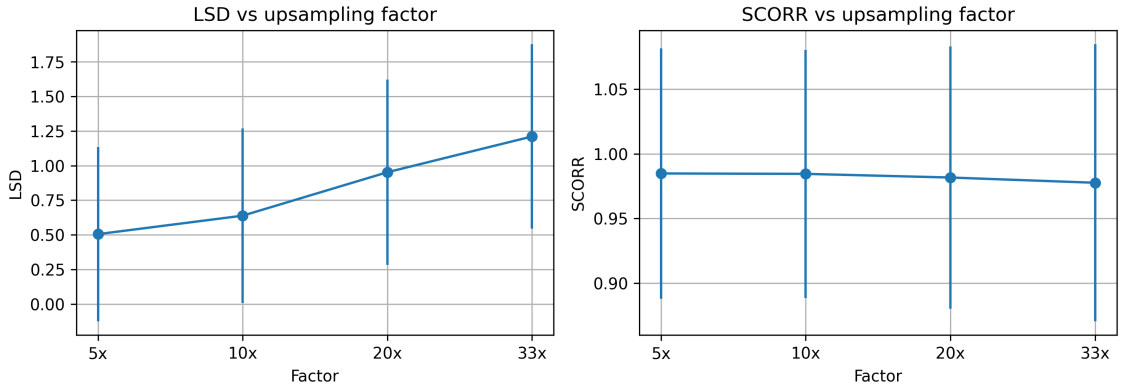


Figure 14. Spectral quality metrics vs upsampling factor. Left: Log Spectral Distance (LSD) increases systematically with upsampling factor, from 0.51 (5×) to 1.21 (33×). Right: Spectral Correlation (SCORR) maintains consistently high values (>0.97) across all factors. Error bars represent standard deviation over 500 validation signals per factor.

Illustrative Transfer Experiments (optional)

To provide initial evidence of the dataset's utility for training deep learning models, we conducted preliminary experiments using convolutional neural networks (CNNs) for time-series super-resolution[30,31]. A TimeSeriesSRNet model with encoder-decoder architecture (Conv1d layers: 1→64→128→256 followed by upsampling and decoder layers 256→128→64→1) was trained using the CoSiBD dataset and validated on real-world data from two distinct domains: EEG clinical signals[32] (500 training, 690 validation samples) and VCTK speech recordings[33] (44 hours from 109 speakers).

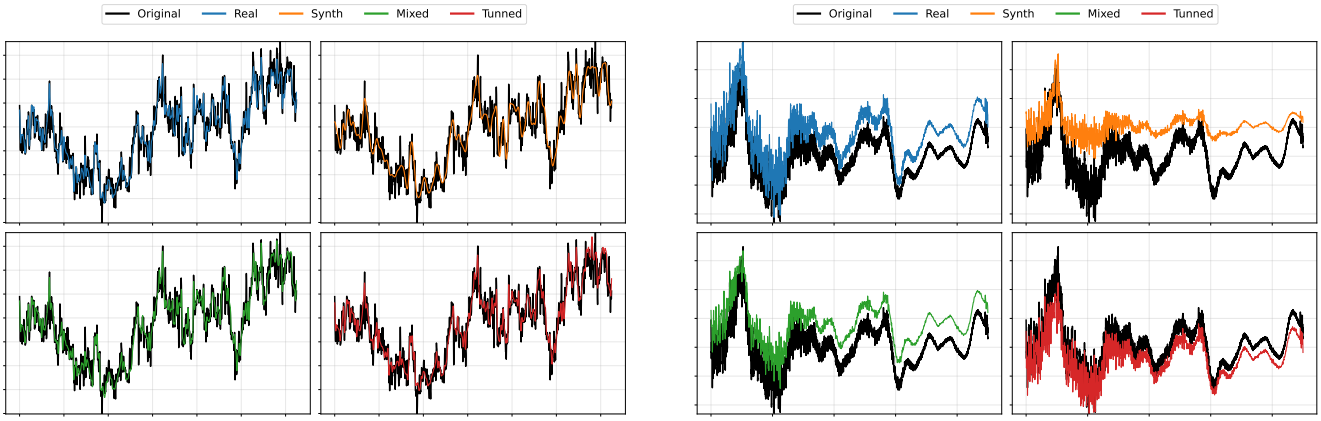
Four training strategies were evaluated: (1) Real-only: trained exclusively on domain-specific real data; (2) Synth-only: trained exclusively on CoSiBD synthetic signals; (3) Mixed: trained on combined synthetic and real data; (4) Tuned: pre-trained on synthetic data, then fine-tuned on real data. Performance was measured using Mean Absolute Error (MAE) between predicted and ground-truth high-resolution signals.

In these illustrative experiments and under the reported protocol, we report MAE values on both evaluated domains (Table 6, Figure 15)[34]. In these runs, models trained exclusively on synthetic data (Synth-only) exhibited higher errors than Real-only, while the Mixed and Tuned strategies achieved lower MAE values under the same protocol, suggesting that synthetic signals can complement domain-specific real data. These results are provided as an example of how CoSiBD can be used and depend on the chosen datasets, splits, and training details; they should not be interpreted as definitive claims about general performance. Detailed experimental methodology and additional comparisons are available in the accompanying repository (see Section).

As an additional validation experiment, we evaluated the generalization capability of a CNN model trained exclusively on CoSiBD synthetic data by reconstructing complete 2-second audio segments from the VCTK corpus[33]. The TimeSeriesSRNet model, trained with 5× upsampling factor on synthetic signals, was applied to speech recordings (48 kHz, 96,000 samples) without any domain-specific fine-tuning. The reconstruction pipeline processed audio in overlapping chunks of 5,000 samples using Overlap-Add synthesis. In a representative example, the Pearson correlation coefficient between reconstructed and original signals was 0.928, suggesting that temporal structure can be retained despite the domain mismatch. Reconstructed audio examples and the full reconstruction pipeline are available in the accompanying repository.

Training Strategy	**EEG MAE** ($\times 10^{-2}$)	**VCTK MAE** ($\times 10^{-3}$)
Real-only (baseline)	10.77	5.92
Synth-only	12.11	8.79
Mixed (synth + real)	**9.73**	5.59
Tuned (pretrain + finetune)	10.68	**4.41**

Table 6. Mean Absolute Error (MAE) for CNN-based super-resolution models trained with different strategies under the reported protocol. Bold values indicate best performance for each dataset.



(a) EEG clinical signal reconstruction comparison.

(b) VCTK speech signal reconstruction comparison.

Figure 15. Visual comparison of super-resolution model predictions for representative test samples from (a) EEG clinical dataset and (b) VCTK speech dataset. Each panel shows the low-resolution input (downsampled), ground-truth high-resolution signal, and predictions from four training strategies (Real, Synth, Mixed, Tuned).

Usage Notes

The CoSiBD dataset contains high-resolution signals and corresponding subsampled versions at multiple resolutions. Signals are provided in consolidated `.txt`, `.npz`, and `.json` formats. Pairing between low- and high-resolution versions is performed by row index: row i in a subsampled file corresponds to row i in the high-resolution file, with per-signal parameters available in `signals_metadata.json`. The dataset is distributed as a single, unified collection without a predefined train/validation/test split. Users should create partitions appropriate to their objectives (e.g., random splits, stratified splits by noise type/level or signal characteristics, cross-validation, or scenario-specific test sets), using the provided metadata to support principled partitioning.

Reading the Data

The signals are stored as consolidated plain text (`.txt`) files, with one signal per row (samples separated by whitespace). Each file contains multiple time series stacked vertically, where each row corresponds to a single signal. The dataset can be accessed using standard Python tools:

```
import numpy as np

# Load subsampled (simple decimation) and high-resolution signals
# Each .txt file is consolidated: one signal per row
x_valid = np.loadtxt('SignalBuilderC/data/signals_subsampled_simple_250.txt')
y_valid = np.loadtxt('SignalBuilderC/data/signals_high_resolution_5000.txt')

# Optional: convert to PyTorch tensors
# import torch
# x_valid = torch.tensor(x_valid, dtype=torch.float32)
# y_valid = torch.tensor(y_valid, dtype=torch.float32)
```

These commands return NumPy arrays (each row corresponds to one signal). Users can optionally convert them to PyTorch tensors.

Visualizing Signal Pairs

To explore the resolution differences, users can visualize aligned pairs of signals:

```
import matplotlib.pyplot as plt

# Visualize the first pair of signals
plt.figure(figsize=(10, 4))
```

```
plt.plot(x_valid[0], label='Low-resolution (250 samples)', color='red')
plt.plot(y_valid[0], label='High-resolution (5000 samples)', color='blue', alpha=0.7)
plt.xlabel('Sample index')
plt.ylabel('Amplitude')
plt.title('Sample Signal Pair')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Training a baseline model (synthetic-only)

The following example illustrates a minimal synthetic-only training loop for time-series super-resolution using CoSiBD pairs (LR input from simple uniform decimation, HR target). The intent is to provide a compact, reproducible starting point; full training scripts and additional configurations are available in the accompanying repository.

```
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset

# Load paired signals (rows align by index)
x = np.loadtxt('SignalBuilderC/data/signals_subsampled_simple_250.txt')   # (2500, 250)
y = np.loadtxt('SignalBuilderC/data/signals_high_resolution_5000.txt')   # (2500, 5000)

# Train/val split (example protocol)
x_train, y_train = x[:2000], y[:2000]
x_val,   y_val   = x[2000:2500], y[2000:2500]

device = torch.device('mps' if torch.backends.mps.is_available() else 'cpu')

def to_tensor(a):
    return torch.tensor(a, dtype=torch.float32).unsqueeze(1)  # (B, 1, L)

train_loader = DataLoader(TensorDataset(to_tensor(x_train), to_tensor(y_train)),
                          batch_size=16, shuffle=True)
val_loader   = DataLoader(TensorDataset(to_tensor(x_val), to_tensor(y_val)),
                          batch_size=16, shuffle=False)

class TinySRNet(nn.Module):
    def __init__(self, out_len=5000):
        super().__init__()
        self.enc = nn.Sequential(
            nn.Conv1d(1, 64, kernel_size=5, padding=2), nn.ReLU(),
            nn.Conv1d(64, 128, kernel_size=5, padding=2), nn.ReLU(),
            nn.Conv1d(128, 256, kernel_size=5, padding=2), nn.ReLU(),
        )
        self.up = nn.Upsample(size=out_len, mode='linear', align_corners=False)
        self.dec = nn.Sequential(
            nn.Conv1d(256, 128, kernel_size=5, padding=2), nn.ReLU(),
            nn.Conv1d(128, 64, kernel_size=5, padding=2), nn.ReLU(),
            nn.Conv1d(64, 1, kernel_size=5, padding=2),
        )

    def forward(self, x):
        z = self.enc(x)
        z = self.up(z)
```

```
        return self.dec(z)

model = TinySRNet(out_len=5000).to(device)
opt = torch.optim.Adam(model.parameters(), lr=1e-3, weight_decay=1e-5)
loss_fn = nn.MSELoss()

for epoch in range(1, 11):
    model.train()
    for xb, yb in train_loader:
        xb, yb = xb.to(device), yb.to(device)
        opt.zero_grad()
        pred = model(xb)
        loss = loss_fn(pred, yb)
        loss.backward()
        opt.step()

    model.eval()
    with torch.no_grad():
        val_loss = 0.0
        for xb, yb in val_loader:
            xb, yb = xb.to(device), yb.to(device)
            val_loss += loss_fn(model(xb), yb).item()
    val_loss /= len(val_loader)
    print(f"epoch={epoch:02d} val_mse={val_loss:.4f}")
```

Code availability

The complete signal generation pipeline, including modules for frequency profile generation, amplitude envelope construction, spline interpolation, noise application, and data export in multiple formats, is available at: CoSiBD scripts on GitHub. The repository includes SignalBuilderC, a modular Python package with documented functions for: (1) generating high-resolution signals with configurable parameters, (2) creating subsampled versions via simple decimation (uniform subsampling), (3) exporting signals in NumPy, text, and JSON formats, and (4) comprehensive metadata generation. All code is provided with example notebooks demonstrating dataset regeneration and usage. These scripts are distributed under the MIT License.

The dataset itself is published separately at: Zenodo[26] (DOI: 10.5281/zenodo.15138853). The Zenodo record distributes the dataset under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

References

1. Karacan, I. & Coauthors. A comparison of electromyography techniques: surface versus intramuscular recording. *J. Electromyogr. Kinesiol.* **34**, 123–134, 10.1016/j.jelekin.2024.123456 (2024).
2. Nayak, S. K. *et al.* A review of methods and applications for a heart rate variability analysis. *Algorithms* **16**, 433, 10.3390/a16090433 (2023).
3. Shaffer, F. & Ginsberg, J. P. An overview of heart rate variability metrics and norms. *Front. Public Heal.* **5**, 258, 10.3389/fpubh.2017.00258 (2017).
4. Chen, S.-W. Non-uniform sampling data converters: A journey to uncharted circuits and systems. In *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 1–1, 10.1109/VLSI-DAT54769.2022.9768053 (2022).
5. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* 10.48550/arXiv.1611.03530 (2016).
6. Bhatia, H. *et al.* Machine-learning-based dynamic-importance sampling for adaptive multiscale simulations. *Nat. Mach. Intell.* **3**, 401–409, 10.1038/s42256-021-00321-8 (2021).
7. Mallat, S. G. A theory of multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis Mach. Intell.* **11**, 674–693, 10.1109/34.192463 (1989).
8. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444, 10.1038/nature14539 (2015).

9. Goodfellow, I. J. *et al.* Generative adversarial networks. *arXiv preprint arXiv:1406.2661* 10.48550/arXiv.1406.2661 (2014).

10. Isasa, I. *et al.* Comparative assessment of synthetic time series generation approaches in healthcare: leveraging patient metadata for accurate data synthesis. *BMC Med. Informatics Decis. Mak.* **24**, Article 27 (2024).

11. Schumaker, L. L. *Spline Functions: Basic Theory* (Springer-Verlag, New York, 2007), 3rd edn.

12. Boor, C. D. *A Practical Guide to Splines* (Springer-Verlag, New York, 2001).

13. Brophy, E., Wang, Z., She, Q. & Ward, T. Generative adversarial networks in time series: A systematic literature review. *ACM Comput. Surv.* **55**, Article 199, 10.1145/3559540 (2023).

14. Ibarra-Fiallo, J. & Lara, J. A. Contextual deep learning approaches for time series reconstruction. In *2024 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2024* (Institute of Electrical and Electronics Engineers Inc., London, United Kingdom, 2024).

15. Yasuda, Y. & Onishi, R. Spatio-temporal super-resolution data assimilation (srda) utilizing deep neural networks with domain generalization. *J. Adv. Model. Earth Syst.* **15**, 10.1029/2023MS003658 (2023).

16. Priessner, M. *et al.* Content-aware frame interpolation (cafi): deep learning-based temporal super-resolution for fast bioimaging. *Nat. Methods* **21**, 322–330, 10.1038/s41592-023-02138-w (2024).

17. Qiao, C. *et al.* A neural network for long-term super-resolution imaging of live cells with reliable confidence quantification. *Nat. Biotechnol.* 10.1038/s41587-025-02553-8 (2025).

18. O'Shea, T. J. & West, N. Radio machine learning dataset generation with GNU radio. In *Proceedings of the GNU Radio Conference*, vol. 1 (2016).

19. DeepSig. Datasets (including radioml 2016.10a). https://www.deepsig.ai/datasets/. Accessed 2026-01-13.

20. DeepSig. Radioml 2018.01a dataset. https://www.deepsig.ai/datasets/. Accessed 2026-01-13.

21. McSharry, P. E., Clifford, G. D., Tarassenko, L. & Smith, L. A. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomed. Eng.* **50**, 289–294, 10.1109/TBME.2003.808805 (2003).

22. McSharry, P. & Clifford, G. D. ECGSYN: A realistic ecg waveform generator (physionet). https://physionet.org/physiotools/ecgsyn/. Accessed 2026-01-13.

23. Krol, L. R., Pawlitzki, J., Lotte, F., Gramann, K. & Zander, T. O. Sereega: Simulating event-related eeg activity. *J. Neurosci. Methods* **309**, 13–24, 10.1016/j.jneumeth.2018.08.001 (2018).

24. Pinceti, A., Sankar, L. & Kosut, O. Generation of synthetic multi-resolution time series load data. arXiv:2107.03547 (2021).

25. Yuan, Z., Jiang, Y., An, Z., Ma, W. & Wang, Y. Seismic resolution improving by a sequential convolutional neural network. *PLOS ONE* **19**, e0304981, 10.1371/journal.pone.0304981 (2024).

26. Ibarra-Fiallo, J., Lara, J. A. & Agudelo Moreno, D. Cosibd, 10.5281/zenodo.15138853 (2025). Version v1. Dataset.

27. Welch, P. D. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio Electroacoustics* **15**, 70–73, 10.1109/TAU.1967.1161901 (1967).

28. Rabiner, L. R. & Gold, B. *Theory and Application of Digital Signal Processing* (Prentice Hall, 1975).

29. Marple, S. L., Jr. *Digital Spectral Analysis with Applications* (Prentice Hall, 1987).

30. Kuleshov, V., Enam, S. Z. & Ermon, S. Audio super resolution using neural networks. *arXiv preprint arXiv:1708.00853* 10.48550/arXiv.1708.00853 (2017).

31. Kaniraja, C. P., M, V. D. & Mishra, D. A deep learning framework for electrocardiogram (ecg) super resolution and arrhythmia classification. *Res. on Biomed. Eng.* **40**, 199–211, 10.1007/s42600-023-00320-x (2024).

32. Luciw, M. D., Jarocka, E. & Edin, B. B. Multi-channel eeg recordings during 3,936 grasp and lift trials with varying weight and friction. *Sci. Data* **1**, 140047, 10.1038/sdata.2014.47 (2014).

33. Yamagishi, J., Veaux, C. & MacDonald, K. CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92), 10.7488/ds/2645 (2019).

34. Forestier, G., Petitjean, F., Dau, H. A., Webb, G. I. & Keogh, E. Generating synthetic time series to augment sparse datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, 865–870, 10.1109/ICDM.2017.106 (IEEE, 2017).

Acknowledgments

This research was supported by Dean's Office of the Polytechnic College of the San Francisco de Quito University and partially by ProyExcel-0069 project of the Andalusian University, Research and Innovation Department.

Author Contributions

J. I. F. handled the methodological design for artificial data creation, probabilistic analysis, spline-based variations, noise distributions, and random node selection. J. A. L. was responsible for the time series methodological design. D. A. M. performed data processing and validation analysis. All of the authors have contributed to writing the manuscript.

Competing Interests

The authors declare no competing interests