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 or telecommunications 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 supporting reproducible research in multi-resolution time-series analysis, particularly deep learning systems, in tasks like temporal super-resolution. 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 describing the signals' segments and documenting all generation parameters, including seeds for full reproducibility. CoSiBD includes diverse signals with non-uniform frequency modulations, capturing gradual transitions and abrupt high-frequency events to reflect a broad range of non-stationary temporal behaviors, and provides both clean and noisy variants. We report a technical validation that includes, among others, a whole study of the application of the proposed dataset for time-series resolution in real-world scenarios.

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. For instance, In biomedical research[?], electroencephalography (EEG) analyses reveal heart function[?,?]. Another example is telecommunications, which rely on signal processing to ensure data fidelity across noisy media[?]. 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. Deep learning approaches, such as Convolutional Neural Networks (CNNs) or Generative Adversarial Networks (GANs) have demonstrated improved performance over traditional techniques in image, speech, and time-series processing tasks[?,?]. These models support fine-grained signal reconstruction, 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. For instance, in medicine there is a limitation by medical privacy regulations such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA)[?]. In other domains, including telecommunications, data availability is limited by proprietary protocols and the high cost of acquiring large-scale channel sounding datasets for diverse environments[?]. 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, for instance, SR can help reconstruct higher-resolution physiological time series (e.g., EEG), potentially improving the analysis of subtle physiological irregularities[?]. SR also applies to audio/speech enhancement and telecommunications, where higher temporal resolution can increase sensitivity to rapid changes and improve signal quality[?].

Deep learning offers adaptive alternatives to these traditional methods. For instance, CNNs are capable of modeling spatio-temporal structure, that are present in real datasets of the above mentioned domains. Preliminary work on synthetic time-series

generation indicates potential for SR[?,?], 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 characteristics of real-world signals. Prior studies have used synthetic data in domains such as biomedical signal analysis[?] and wireless communications[?], 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. As it is designed to resemble real-world signals, our dataset is intended with a double purpose: a) to provide a resource for training and evaluating deep learning SR models under controlled, reproducible conditions, which can constitute a sort of benchmark for this problem; and b) a resource for training deep learning models to be used for SR of real-world signals (either directly or finetuning them with the real data), particularly in scenarios where real signals are scarce and not enough for a complete training of those models. 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 previously used in research presented at the International Conference on Signal Processing and Machine Learning[?], with good prelimary results for signal reconstruction using deep learning. CoSiBD is made available to support further development in deep learning approaches for temporal super-resolution.

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[?,?]	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[?]	Wireless communications (I/Q)	Dataset	No (classification benchmark)	N/A (not SR)	Simulated channel effects + SNR variability	Dataset-level; not SR-paired
ECGSYN[?,?]	ECG (physiology)	Simulator/tool	Configurable	Configurable (via sampling settings)	Model-based; supports controlled variability	Configurable via simulator parameters (user-defined)
SEREEGA[?]	EEG (physiology)	Simulator/ toolbox	Configurable	Configurable (user-defined)	Supports noise and event-related components	Configurable via simulator parameters (user-defined)
LoadGAN[?]	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)[?]	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.

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

Related synthetic time-series resources

Publicly available synthetic resources for temporal signals exist, but they are typically designed for tasks other than time-series 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 (Signal-to-noise ratio) and channel impairments, mainly to benchmark automatic modulation classification rather than paired SR reconstruction[?,?,?]. In biomedical signal processing, physiological simulators such as ECGSYN (electrocardiography) and SEREEGA (EEG) enable controlled generation with tunable morphology, sampling settings, and noise, supporting method development when real data access is constrained[?,?,?]. 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[?]. 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[?].

Table **??** summarizes these representative resources. Columns indicate the primary **Domain**, the **Form** of distribution (fixed dataset vs. generator), and the specific **Noise** models included. We explicitly check for **Paired LR–HR SR** capability (Low Resolution - High Resolution); resources marked as "Configurable" in this column allow users to generate pairs by running simulators at different settings, but typically do not distribute a standardized, fixed benchmark for SR. The **Reproducibility granularity** column notes whether exact regeneration is supported at the sample level. CoSiBD is designed to close the gap by providing a fixed, standardized set of LR–HR pairs with explicit nuisance modeling (noise and structured interference) and comprehensive metadata, enabling reproducible benchmarking across multiple difficulty levels (defined by varying downsampling factors and noise intensities).

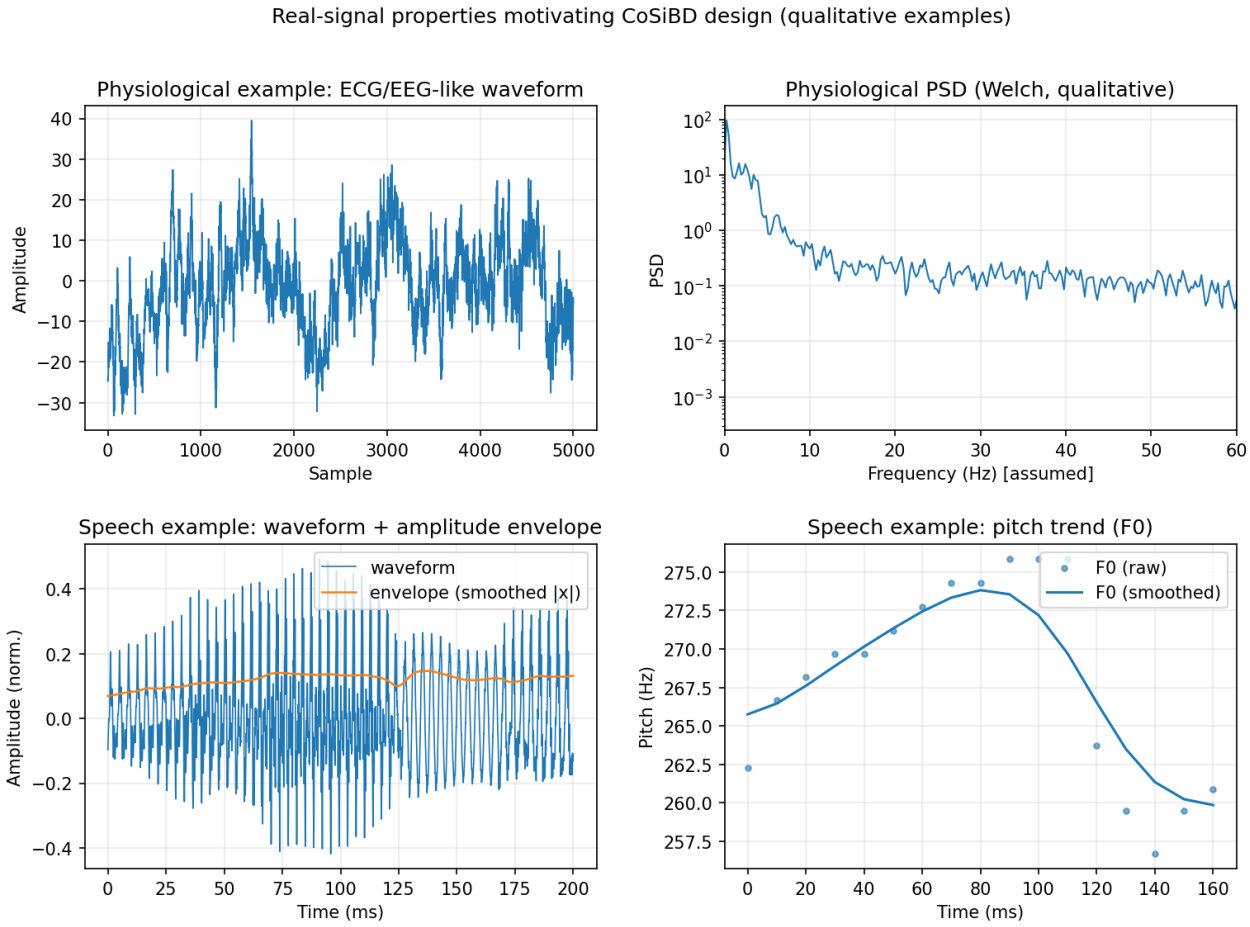


Figure 1. Qualitative real-signal properties motivating the CoSiBD design. The physiological example illustrates non-stationarity in the waveform and structured spectral content; the speech example illustrates amplitude-envelope dynamics and a smoothly varying pitch (F0) trend. These observations motivate CoSiBD mechanisms such as regime partitioning with change-points, low/high-frequency bands, and spline-based envelopes/frequency profiles.

Methods

The methodology used to generate the synthetic temporal signals that constitute the CoSiBD dataset is illustrated in Figure **??**, and will be explained later.

Design rationale inspired by real signals. It is important to note that one of the main applications of the proposed dataset will be the training of deep learning models to be used for SR purposes in other real-world datasets, like, for instance, physiological or speech signals. Therefore, there is a need for our dataset to resemble real-world data. In particular, real signals exhibit (i) non-stationary regime changes, (ii) coexisting low- and high-frequency components with 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 **??** provides qualitative examples of these motivating properties found in real-world time series; the main goal of our dataset is to be able to capture challenging structure for SR benchmarking rather than match a specific domain distribution.

Figure **??** presents four subfigures labeled A–D to illustrate these concepts. Figure **??**A (top left) shows a real EEG segment (source:[?]) illustrating non-stationary oscillatory bursts. Figure **??**B (top right) displays the corresponding spectrogram, highlighting structured spectral content. Figure **??**C (bottom left) presents a speech signal segment (source: VCTK corpus[?]) showing clear amplitude-envelope dynamics. Figure **??**D (bottom right) shows the speech spectrogram, revealing a smoothly varying pitch (F0) trend.

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 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: (1) Additive white Gaussian noise (AWGN) with configurable standard deviation (relative to signal RMS amplitude), representing broadband sensor thermal noise; and (2) structured sinusoidal noise bursts (deterministic sinusoidal components), representing narrow-band interference such as powerline hum (50/60 Hz). Noise is applied probabilistically with a fixed 50% probability per signal. When noise is present, the specific parameters (type, amplitude, frequency for structured noise) are recorded in the metadata, allowing users to benchmark denoising or super-resolution under specific degradation conditions.

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 **??** illustrates this qualitative motivation through four panels (A–D). Panel A displays a synthetic wideband noise floor. Panel B adds the structured narrow-band component. Panel C shows the combined time-domain signal, and Panel D presents the frequency spectrum, clearly showing the powerline-like artifact. This explicitly models the periodic contamination observed in real recordings; 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 reported Hz values while preserving the underlying discrete sequences, which is a key feature of CoSiBD's design.

Figure **??** clarifies this convention using six panels (A–F). Panels A–C illustrate the time-domain representation: A shows the raw discrete sequence against sample index; B shows the same sequence mapped to a duration T_1; and C mapped to T_2, demonstrating that the sample values are invariant. Panels D–F show the corresponding frequency-domain effects: D is the normalized spectrum (cycles/sample), while E and F show the Hz mappings for T_1 and T_2 respectively, illustrating how the physical frequency interpretation scales with the assumed duration.

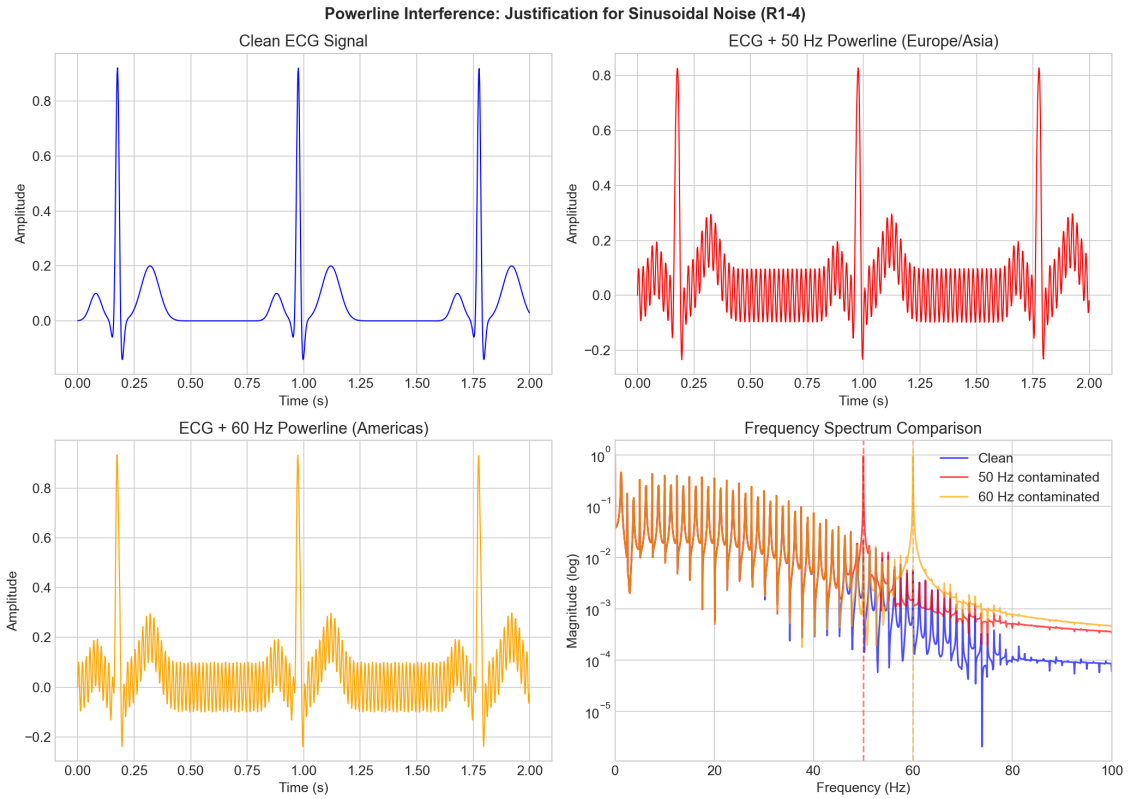


Figure 2. Qualitative motivation for the structured interference term used in CoSiBD. An illustrative example shows how adding a narrow-band sinusoidal component (interpretable as 50/60 Hz under the illustrative convention $T = 4\pi$ s) produces the characteristic periodic contamination observed in real recordings, while broadband noise captures the measurement floor.

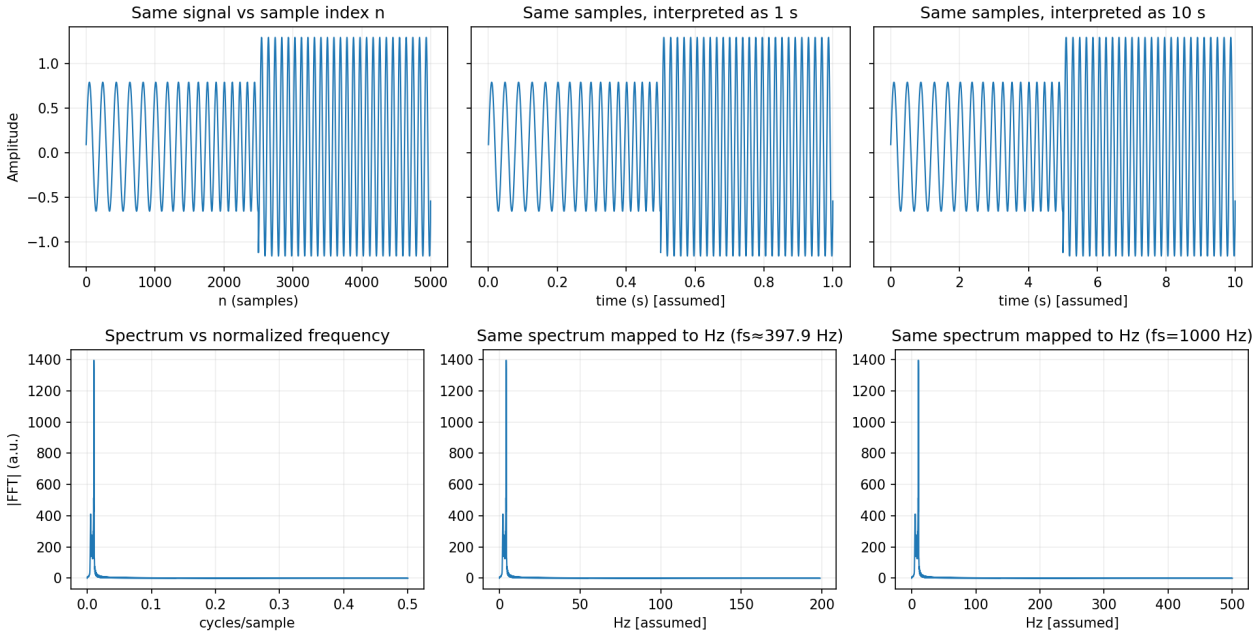


Figure 3. Sampling/unit convention in CoSiBD. Top: the same discrete sequence $x[n]$ can be plotted against the sample index or under different assumed time scalings. Bottom: the intrinsic frequency axis is normalized (cycles/sample); mapping to "Hz" depends on the assumed sampling rate f_s (two example mappings shown).

Data Records

The Complex Signal Benchmark Dataset (CoSiBD) is publicly available on Zenodo[?] and consists of synthetic temporal signals, mainly created to support the development and evaluation of temporal super-resolution (SR) algorithms, and also to train deep learning models that can be used for SR in real-world signals. This section provides an overview of the dataset structure, content, and storage format, as well as the parameters that rule the generation of data and the metadata that enrich our dataset.

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 $T = [0, 4\pi]$ (s), which, under the illustrative convention used, corresponds to $f_s = 5000/(4\pi) \approx 398$ Hz. 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 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 (illustrative $f_s \approx 11.9$ Hz for $T = 4\pi$ s), 250 (illustrative $f_s \approx 19.9$ Hz for $T = 4\pi$ s), 500 (illustrative $f_s \approx 39.8$ Hz for $T = 4\pi$ s), and 1,000 samples (illustrative $f_s \approx 79.6$ Hz for $T = 4\pi$ s). These low-resolution versions serve as inputs for SR benchmarking against the original 5,000-sample target. Stored in .npz, .txt, and .json formats.

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 (described later in detail) and configuration are stored in `signals_metadata.json` (per-signal metadata, one entry per signal), `signals_metadata_consolidated_2500.json`, and `dataset_summary.json`.

Regarding the three formats used for both high-resolution and subsampled signals, we provide here some additional information for each format: (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.

Reproducibility is ensured through documented 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 (described later in detail) 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.

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 **??** 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`	String	"signal_0000"	Unique identifier for the generated signal.
`index`	Integer	0	Numeric index of the signal (0 to N-1).
`seed`	Integer	10000	Seed used for random generation of this specific signal.
`t_start, t_end`	Float	0.0, 12.56	Start and end time of the signal domain.
`base_points`	Array	[[0.0, 2.07]...]	Control points (t, f) for the frequency spline.
`variation_type`	List[Str]	["low", "low"]	Labels for each interval: "low", "high", "no_change".
`tau_frequency`	Float	1.15	Tension parameter for the frequency spline.
`amplitude_values`	Array	[0.72, 1.22]	Values of the amplitude envelope at knots.
`noise_profile`	Dict	{...}	Parameters for added additive noise (if any).

Table 2. Metadata schema describing the JSON structure for each signal. Key fields define the frequency trajectory (`base_points`, `variation_type`) and amplitude envelope, allowing precise reconstruction or filtering.

Table **??** details the key fields describing each signal, specifically the `variation_type` list which labels each segment (e.g., “low”, “high”, or “no_change”), enabling users to filter for signals containing specific dynamic behaviors. The following JSON snippet corresponds to a signal generated with seed 10000. It shows a signal composed of 2 intervals (indicated by `base_points`), where the frequency profile is modulated by a spline (`tau_frequency`=1.15) and amplitude follows a zero-order hold model. The `variation_type` array `['low', 'low', ...]` indicates the nature of frequency content in each interval.


```
{
    "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
}
```

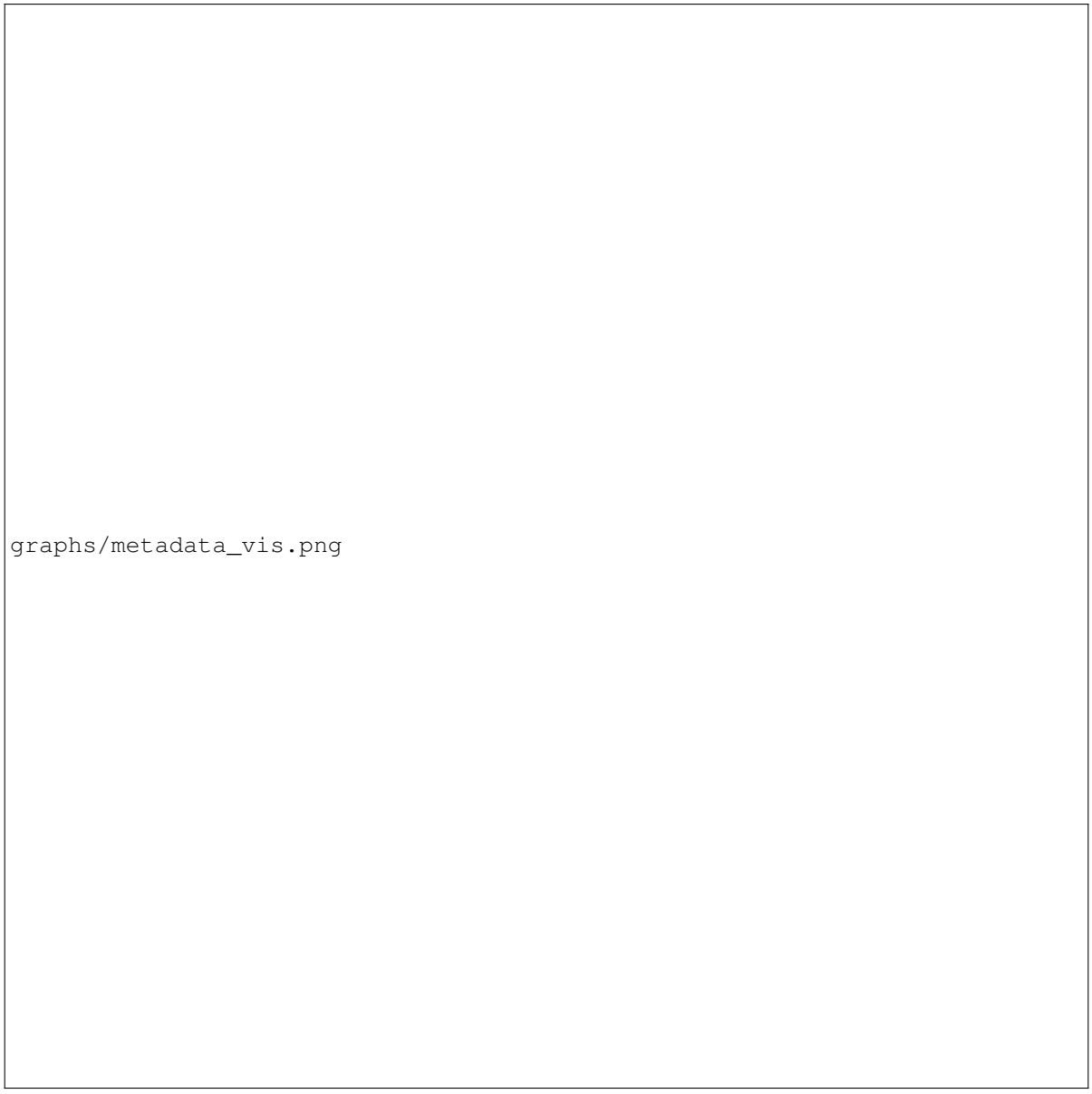


Figure 4. Visual representation of the metadata for signal_0000 (Example 1). The plot illustrates how `base_points` define temporal intervals and frequency targets, while `amp_values` control the amplitude envelope segments, corresponding directly to the JSON structure.

Parameters for signal generation

As we anticipated before, our dataset is generated based on some parameters, that are outlined in Table **??**. Each high-resolution signal was generated with a unique seed (10,000–12,499) and sampled parameter values within the defined ranges, supporting diversity while maintaining reproducibility.

Parameter	Range / Options	Default	Description
Low Frequency	1–5 Hz	Random	Low-frequency component ($T = 4\pi$ s convention).
High Frequency	20–100 Hz	Random	High-frequency variations for transitions.
Change Points	2–11	Random	Number of frequency transitions per signal.
Change Locations	Continuous	Random	Time locations where transitions occur.
Variation Type	{low, high, no_change}	Balanced	Category of frequency change per segment.
Amplitude Range	3–16	Random	Bounds for amplitude envelope generation.
Vertical Offset	$N(0, 3.0)$	0.0	Normally distributed offset added to signals.
Spline Type	Zero-Order (70%), Tension (30%)	-	Interpolation method for envelopes.
Tension (freq)	[1, 2]	1.5	Tension parameter for frequency splines.
Tension (amp)	{1, 3, ..., 20}	Random	Tension parameter for amplitude splines.
Noise Prob.	0.0–1.0	0.5	Probability of adding noise to a signal.
Seed	Integer	-	Unique initialization seed per signal.

Table 3. Configuration parameters used to generate the dataset. Ranges define the sampling space for the 2,500 signals, ensuring diversity. Default values apply when a parameter is not randomized.

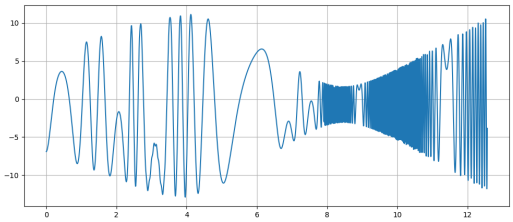
To explicitly characterize dataset diversity and complexity, CoSiBD spans multiple controlled axes of variation (Table **??**), including the number and location of change points, categorical transition types, low/high frequency bands, and amplitude-envelope configurations. The resulting variability is visible in representative realizations (Figures **??** and **??**) and is quantified in Technical Validation via the distribution of dominant frequencies (Figure **??** and Table **??**) and PSD behavior under different resolutions and noise settings (Figures **??** and **??**). 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 **??** 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.

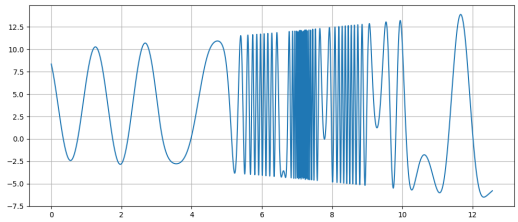
Figure **??** displays four additional synthetic signals generated using different configuration parameters. These examples demonstrate the variability in temporal structure across instances in the dataset.

Custom Dataset Generation

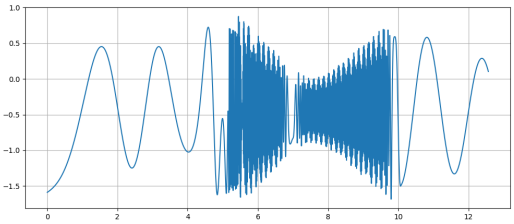
In addition to the pre-generated dataset, the CoSiBD package includes a command-line interface (CLI) that allows users to generate custom datasets with their own parameter distributions. Figure **??** demonstrates the usage of this tool, showing the command syntax for specifying signal length, number of samples, and noise probability, along with the resulting output logs confirming generation.



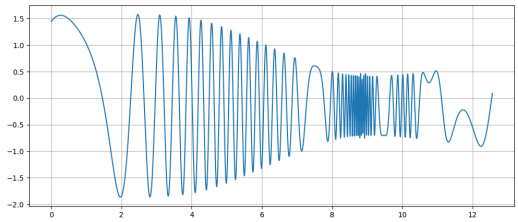
(a) High-resolution signal (5000 samples).



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

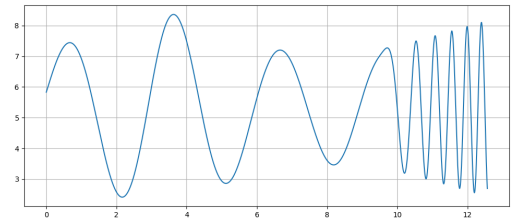


(c) Low-resolution signal (250 samples).

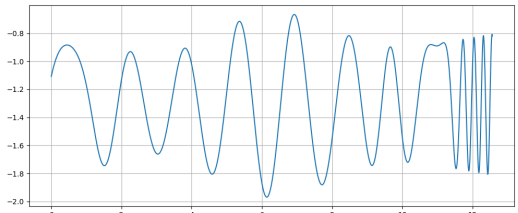


(d) Signal with added noise.

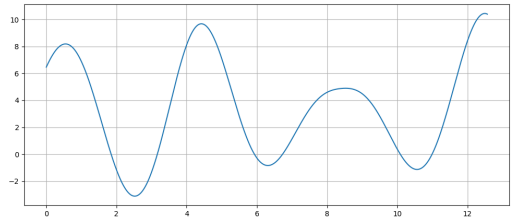
Figure 5. A synthetic signal sampled at different resolutions: (a) high (5000 samples), (b) medium (500 samples), (c) low (250 samples), and (d) with added noise. These examples reflect the multi-resolution and noise conditions present in the dataset.



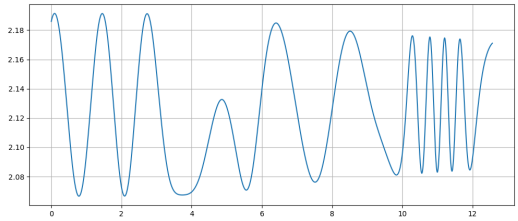
(a) Signal with increasing frequency over time.



(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.

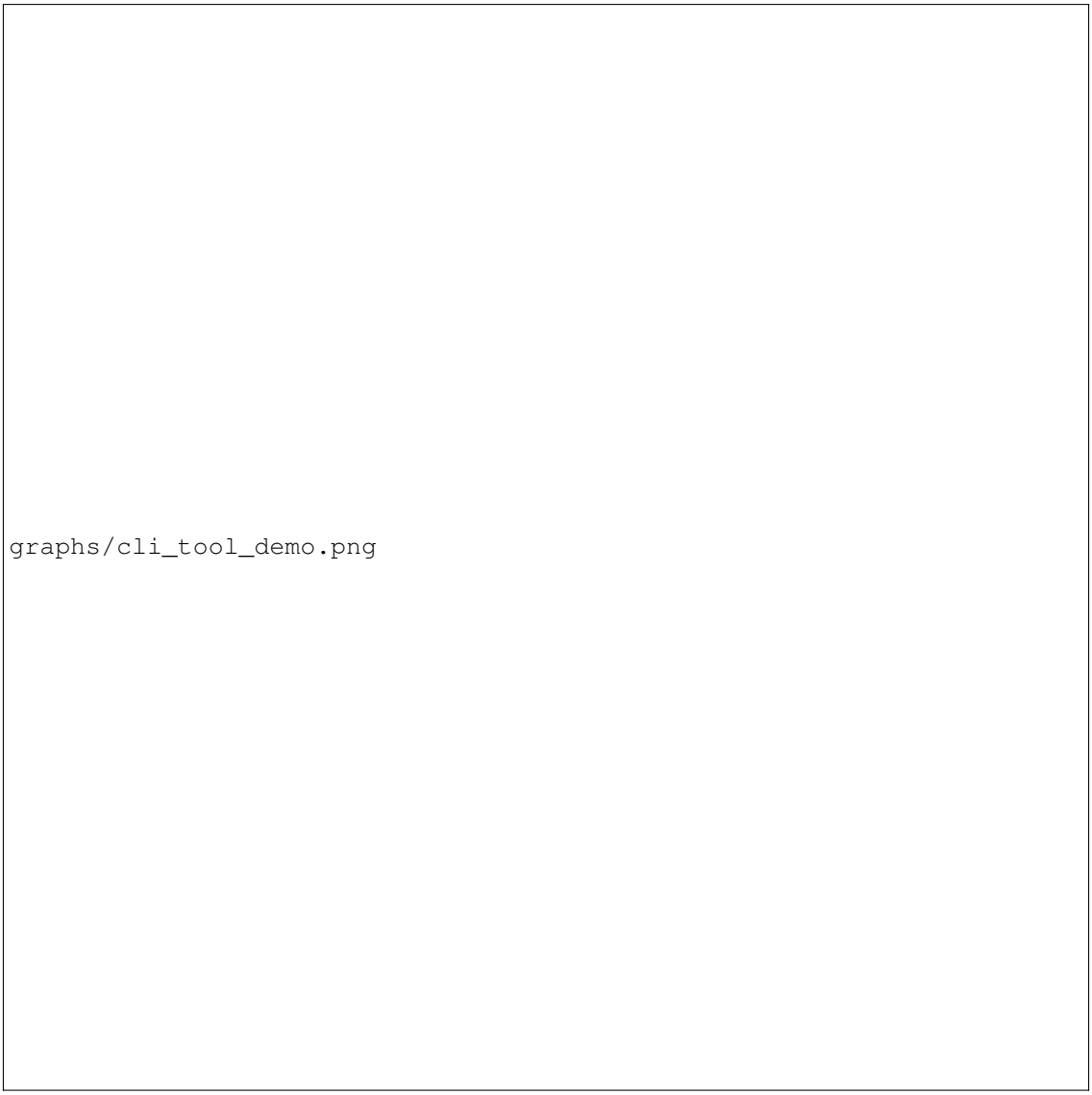


Figure 7. Demonstration of the CoSiBD Command Line Interface (CLI). Users can generate new dataset instances by specifying parameters such as signal count, sampling resolution, and noise probability directly from the terminal.

The full dataset is hosted in Zenodo[?] (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. Additionally, we provide a multi-scale super-resolution benchmark to demonstrate the dataset's utility in training deep learning models, and illustrative transfer learning experiments using real-world EEG and speech data. 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 ($n = 50$ for spectral analysis, $n = 2500$ for benchmarks) provides a compact but informative sample to summarize variability. Sampling resolutions (ranging from 150 to 5000 samples) reflect scenarios

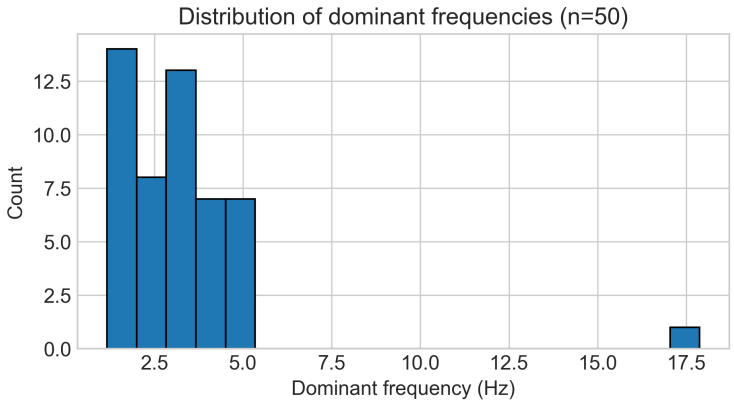


Figure 8. Distribution of dominant frequencies in 50 independently generated signals (reported in Hz, assuming the illustrative convention $T = 4\pi$ s; for other time domains, the axis rescales by $4\pi/T$).

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.

requiring different levels of detail, aligning with typical signal processing use cases. Noise amplitudes (Gaussian noise with $\sigma \in [0.0, 0.2]$) were motivated by common acquisition artifacts, with the goal of providing a controllable benchmark rather than an exhaustive model of any specific measurement pipeline.

Analysis of Dominant Frequency Distribution

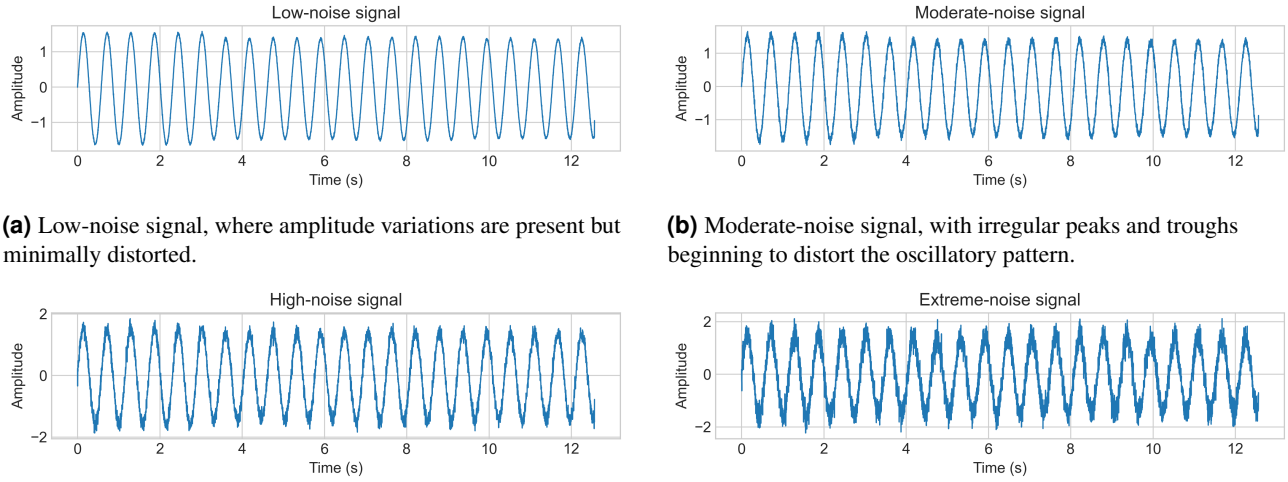
To assess the stability and variability of the primary spectral components, we analyzed the distribution of dominant frequencies across multiple generated signals. A total of fifty independent signals were synthesized using identical input parameters. To examine their spectral characteristics, we computed 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[?]. 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.

The results, shown in Figure **??** and Table **??**, 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. This behavior confirms the method's ability to generate signals with consistent primary structures while introducing controlled variability, a key requirement for robust machine learning training sets[?].

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



(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 9. 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[?]. 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[?].

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.

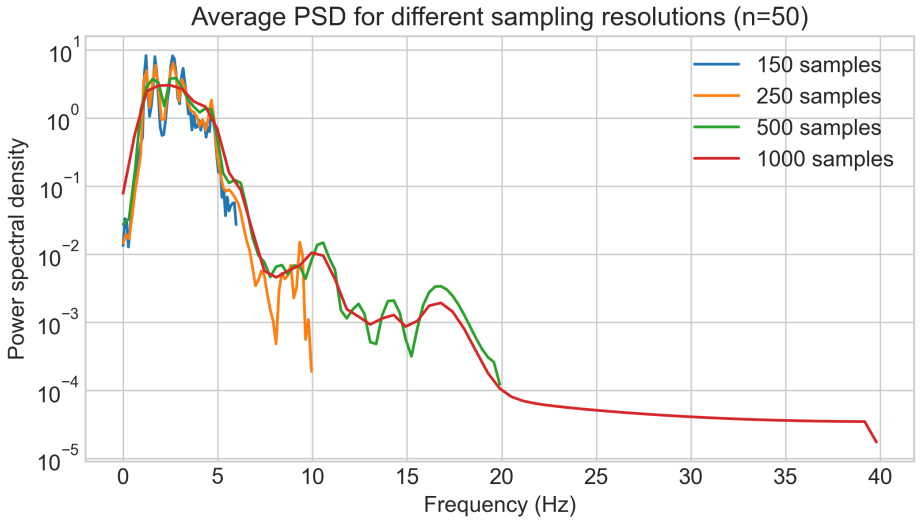


Figure 10. Average power spectral density (PSD) for different sampling resolutions based on 50 independent runs (Hz axis under the illustrative convention $T = 4\pi$ s).

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

samples), exhibit a noticeable reduction in detail within the higher-frequency range. These lower-resolution curves display greater fluctuations, consistent with the theoretical effects of subsampling and aliasing[?]. In contrast, the higher sampling resolutions (500, 1000 samples) demonstrate a smoother and more stable spectral profile. This analysis confirms that while lower sampling rates introduce aliasing artifacts, the dataset provides spectral fidelity comparable to theoretical expectations when sufficient resolution is employed.

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.

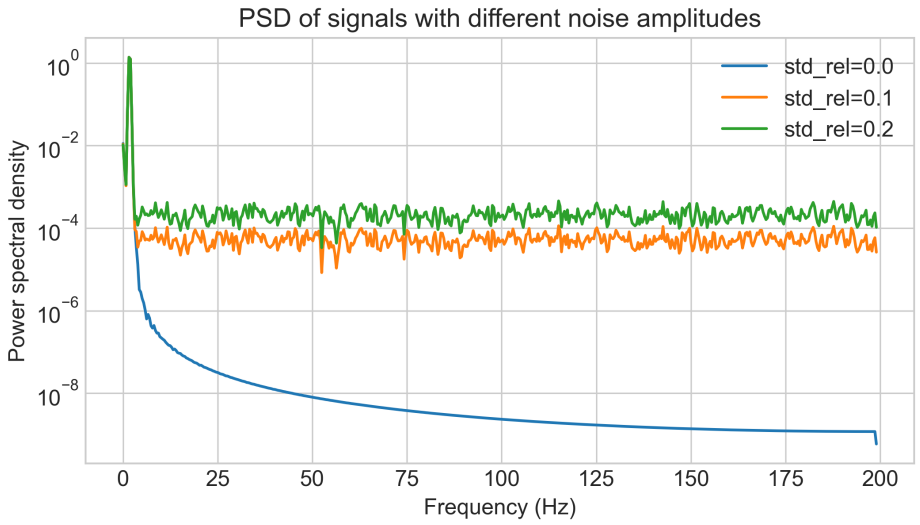


Figure 11. Power spectral density (PSD) of signals generated with different noise amplitudes (Hz axis under the illustrative convention $T = 4\pi$ s).

Figure **??** illustrates the impact of different noise amplitudes on the Power Spectral Density (PSD) under the reported settings. As the noise amplitude increases—from 0.0 (blue curve) to 0.2 (green curve)—the estimated PSD exhibits increased variability at higher frequencies, while the low-frequency region remains comparatively stable.

Across these settings, the low-frequency region changes less than the higher-frequency region. This stability suggests that CoSiBD signals retain their primary structural characteristics even under significant noise, a critical property for robust representation learning[?].

Multi-Scale Super-Resolution Benchmark

To illustrate a baseline use case of CoSiBD, we trained a series of convolutional neural network (CNN) models for time series super-resolution at four different scaling factors: 5× (1000→5000), 10× (500→5000), 20× (250→5000), and 33× (150→5000). All models employed the TimeSeriesSRNet architecture—a five-layer encoder-decoder network with 1D convolutional layers and bilinear upsampling, inspired by deep residual architectures for audio generation[?]. For this benchmark, the 2,500 high-resolution signals were partitioned into an experiment-specific split of 2,000 paired signals for training and 500 held-out signals for validation.

Each model was trained using mean squared error (MSE) loss, a standard objective for regression tasks requiring broad mode coverage[?], using the Adam optimizer (learning rate 0.001) and early stopping. Training was conducted on Apple Silicon GPU (MPS backend) to accelerate convergence.

Table **??** summarizes the validation performance. In these runs, validation loss increased systematically with upsampling factor, reflecting the inherent difficulty of reconstructing fine temporal details from severely undersampled inputs (Table **??**, Figure **??**).

To complement amplitude-based validation, we computed spectral fidelity metrics. Log Spectral Distance (LSD) increased from 0.51 (5×) to 1.21 (33×), while Spectral Correlation (SCORR) remained consistently high (Table **??**, Figure **??**). Figure **??** presents representative spectrogram comparisons across all upsampling factors.

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, while SCORR (Spectral Correlation) measures frequency-domain similarity. All models completed the full 50-epoch training without early termination, showing stable convergence.

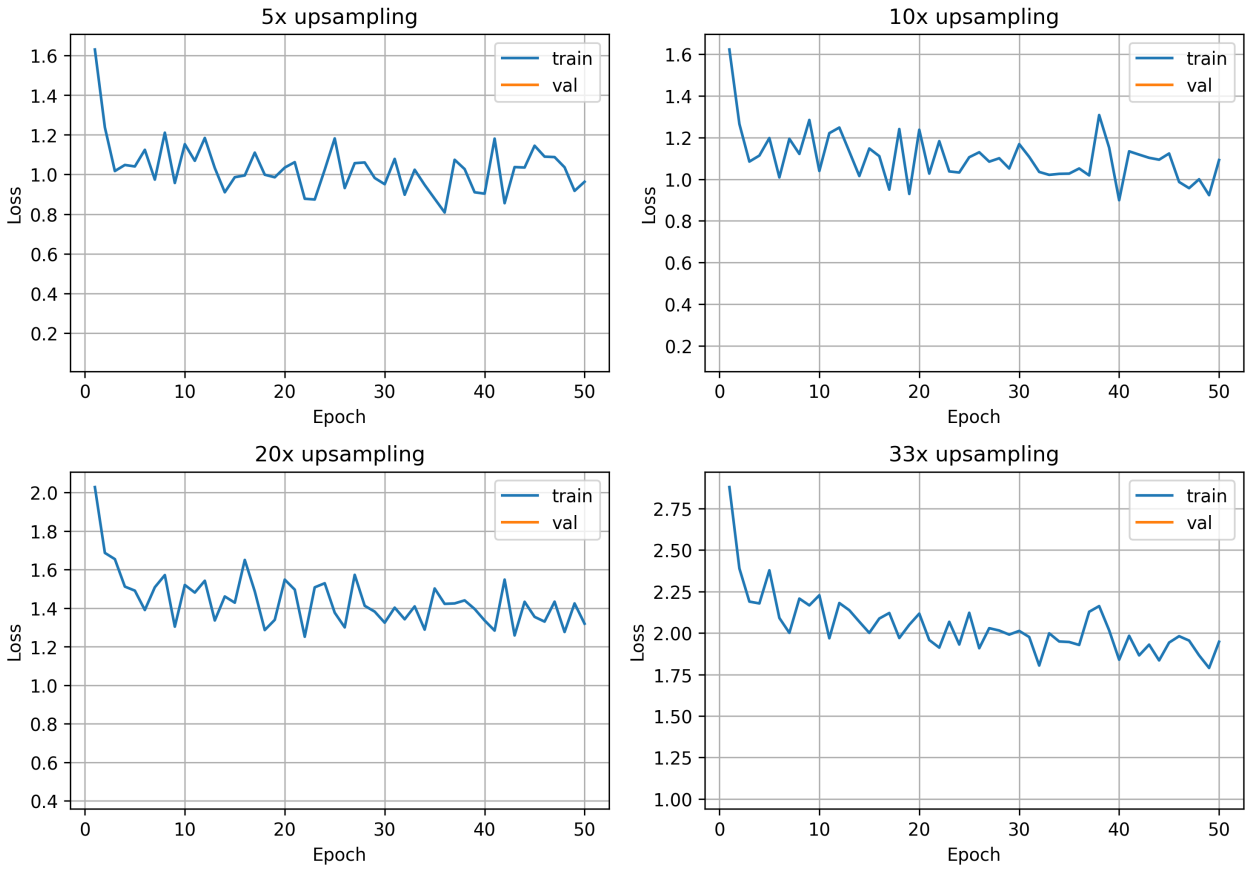


Figure 12. Training and validation loss evolution across all four upsampling factors (5×, 10×, 20×, 33×). Each panel shows loss curves during training; in these runs, training and validation curves follow similar trends without pronounced divergence. The systematic increase in final validation loss with upsampling factor reflects the inherent difficulty of reconstructing fine temporal details from severely undersampled inputs.

Figure **??** illustrates the training and validation loss evolution for all four upsampling factors. Representative prediction examples (Figure **??**) provide qualitative comparisons of reconstructed outputs against ground truth across scaling factors.

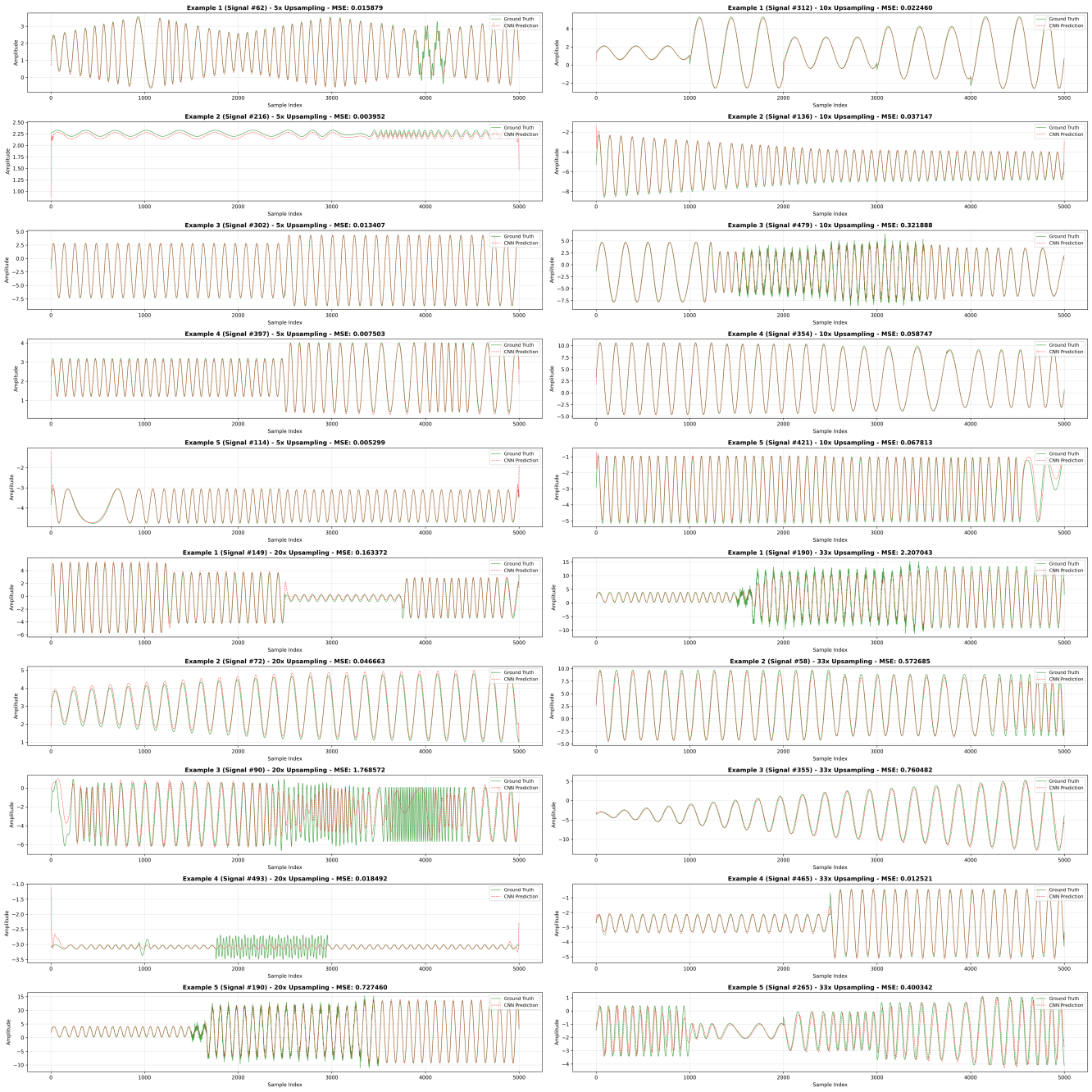


Figure 13. Representative prediction examples across all upsampling factors. Each quadrant shows prediction comparisons for a different scaling factor (5×, 10×, 20×, 33×), displaying low-resolution inputs, ground-truth high-resolution signals, and CNN-reconstructed outputs.

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.

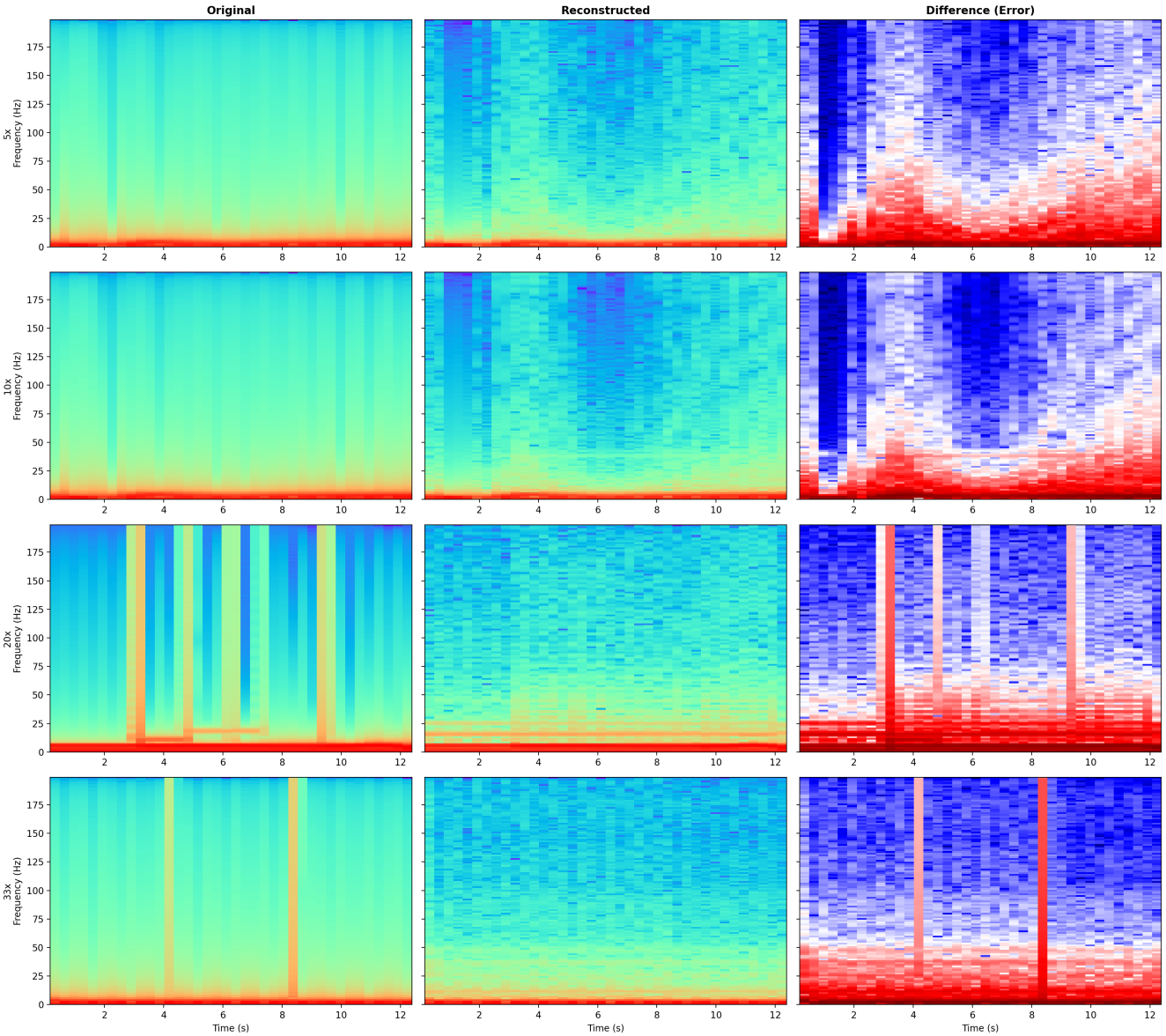


Figure 14. Spectrogram comparison across all upsampling factors. Each row represents a different upsampling factor (5×, 10×, 20×, 33×), showing original signal (left), CNN-reconstructed signal (center), and spectral difference (right). Reconstruction artifacts become more visible at higher upsampling rates. Representative signals selected based on median Log Spectral Distance (LSD) for each factor.

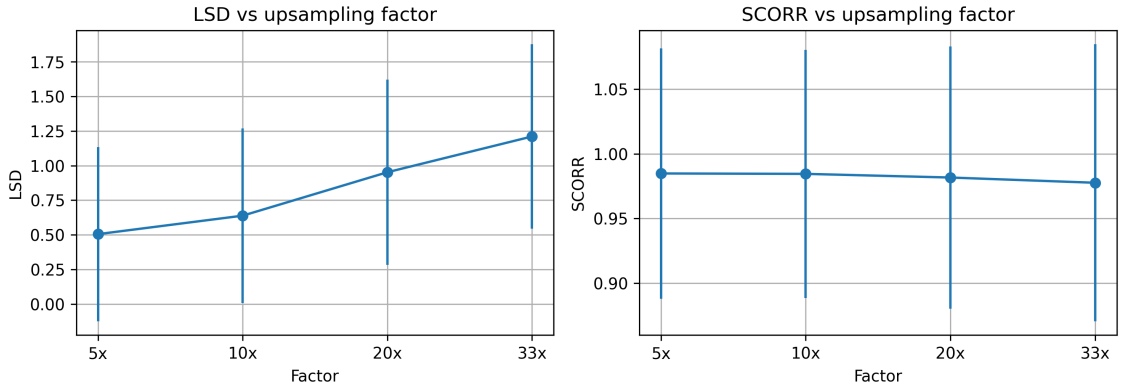


Figure 15. Spectral quality metrics vs upsampling factor. Left: Log Spectral Distance (LSD) increases systematically with upsampling factor, from 0.51 (5×) to 1.21 (33×). Right: Spectral Correlation (SCORR) maintains consistently high values (>0.97) across all factors. Error bars represent standard deviation over 500 validation signals per factor.

Illustrative Transfer Learning Experiments

To demonstrate the practical utility of CoSiBD for training deep learning models, we conducted transfer learning experiments using convolutional neural networks (CNNs) for time-series super-resolution[?,?]. A TimeSeriesSRNet model (encoder-decoder architecture with 1D convolutions: 1→64→128→256, bilinear upsampling, decoder 256→128→64→1) was evaluated on two distinct real-world domains: EEG clinical signals[?] (500 training, 690 validation samples) and VCTK speech recordings[?] (44 hours from 109 speakers).

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

The results (Table **??**, Figure **??**) demonstrate that integrating CoSiBD improves model performance compared to using limited real data alone. While models trained only on synthetic data (Synth-only) yielded higher errors due to domain shift, the **Mixed** and **Tuned** strategies consistently outperformed the Real-only baseline. Specifically, fine-tuning a CoSiBD-pretrained model reduced error significantly on the VCTK dataset, suggesting that the synthetic dataset effectively captures universal temporal structures relevant to super-resolution tasks.

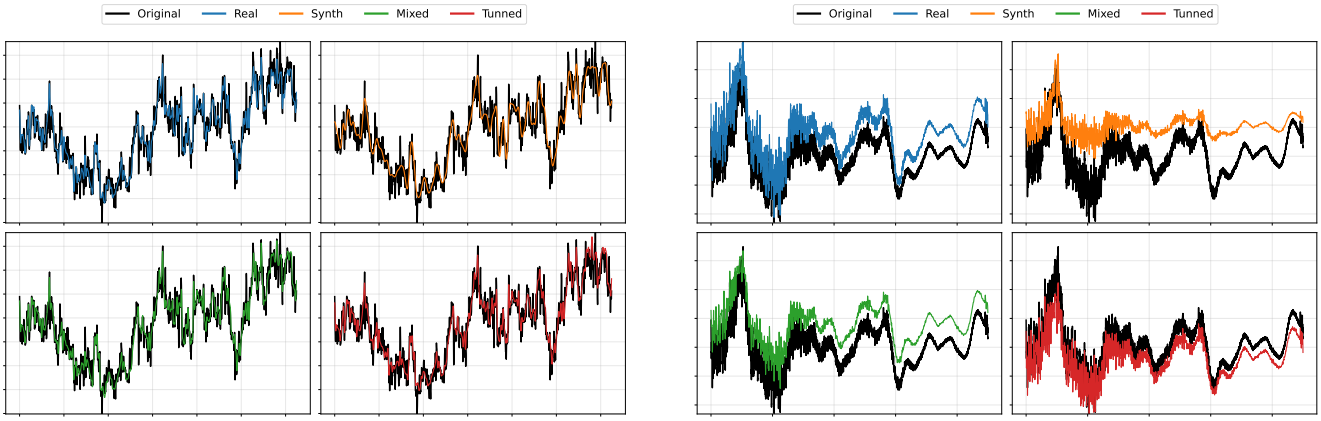
Training Strategy	**EEG MAE** ($\times 10^{-2}$)	**VCTK MAE** ($\times 10^{-3}$)
Real-only (baseline)	10.77	5.92
Synth-only	12.11	8.79
Mixed (synth + real)	**9.73**	5.59
Tuned (pretrain + finetune)	10.68	**4.41**

Table 6. Mean Absolute Error (MAE) for CNN-based super-resolution models. Embedding CoSiBD data (Mixed and Tuned strategies) improves reconstruction accuracy compared to training on limited real data alone.

As a further validation of structural transferability, we applied the TimeSeriesSRNet trained solely on CoSiBD (5× upsampling) to reconstruct full 2-second audio clips from the VCTK corpus without any fine-tuning. Despite the domain gap, the model successfully recovered intelligible speech with preserved harmonic structure (mean Pearson correlation $r = 0.928$). This indicates that CoSiBD's diverse frequency and envelope parameters generalize well to complex, non-stationary signals like human speech.

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



(a) EEG clinical signal reconstruction.

(b) VCTK speech signal reconstruction.

Figure 16. Visual comparison of super-resolution predictions. CoSiBD-enhanced models (Mixed/Tuned) recover finer details in both (a) EEG transients and (b) speech waveforms compared to the baseline.

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[?] (DOI: 10.5281/zenodo.15138853). The Zenodo record distributes the dataset under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

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