Time Series Embedding Methods for Classification Tasks: A Review

Habib Irani, Yasamin Ghahremani, Arshia Kermani, Vangelis Metsis*

{*habibirani, zub11, arshia.kermani, vmetsis*}*@txstate.edu*

[a]*Department of Computer Science, Texas State University, San Marcos, 78666, TX, USA*

Abstract

Time series analysis has become crucial in various fields, from engineering and finance to healthcare and social sciences. Due to their multidimensional nature, time series often need to be embedded into a fixed-dimensional feature space to enable processing with various machine learning algorithms. In this paper, we present a comprehensive review and quantitative evaluation of time series embedding methods for effective representations in machine learning and deep learning models. We introduce a taxonomy of embedding techniques, categorizing them based on their theoretical foundations and application contexts. Our work provides a quantitative evaluation of representative methods from each category by assessing their performance on downstream classification tasks across diverse real-world datasets. Our experimental results demonstrate that the performance of embedding methods varies significantly depending on the dataset and classification algorithm used, highlighting the importance of careful model selection and extensive experimentation for specific applications. This study contributes to the field by offering a systematic comparison of time series embedding techniques, guiding practitioners in selecting appropriate methods for their specific applications, and providing a foundation for future advancements in time series analysis. To facilitate further research and practical applications, we provide an open-source code repository implementing these embedding methods: `https://github.com/imics-lab/time-series-embedding`.

Keywords: time series embedding, dimensionality reduction, feature extraction, classification, machine learning, deep learning, signal processing
2020 MSC: 62M10, 68T07, 62H30, 37M10

*Corresponding author
Email address: `vmetsis@txstate.edu` (Vangelis Metsis)

1. Introduction

Time series embedding is a technique used to represent time series data in the form of vector embeddings, also known as feature vectors. Today, time series analysis methods have emerged as a fundamental element across a vast amount of applications ranging from finance, as in the work of Zhu and Huang (2022), to healthcare, as demonstrated in Nejedly et al. (2022); Morid et al. (2023); Chen et al. (2021); Lee and Hauskrecht (2021); Soenksen et al. (2022), engineering applications such as machine health monitoring, predictive maintenance, and fault detection, proposed by Zhao et al. (2019); Li et al. (2020), and social sciences, explored by Santosh et al. (2018). In this review, we will primarily use the term 'time series.' However, the term 'signal' will also be used, particularly when discussing data from domains where it is the conventional term (e.g., bioelectrical signals, mechanical system signals), and is considered synonymous with 'time series' for the purpose of this work.

As machine learning and deep learning techniques continue to advance, there is a growing need for effective methods to represent and analyze time series data in these models. Tasks such as anomaly detection, classification, pattern recognition, prediction, and decision-making now heavily rely on robust methods that could accurately embed these often high-dimensional data into scalable yet informative representations. The importance of studying and evaluating different time series embedding methods stems from several key factors:

- *Dimensionality reduction*: Time series data often has high dimensionality, which can lead to computational challenges and the curse of dimensionality. Effective embedding methods can reduce the dimensionality while preserving essential temporal patterns and relationships.

- *Feature extraction*: Embeddings can automatically extract relevant features from raw time series data, potentially capturing complex temporal dependencies that may not be apparent in the original representation.

- *Improved model performance*: Well-designed embeddings can lead to significant improvements in the performance of downstream machine learning tasks, such as classification, clustering, and forecasting.

- *Transfer learning*: Embeddings learned from large datasets can be transferred to smaller, related datasets, enabling more effective learning in scenarios with limited data.

- *Interpretability*: Some embedding methods can provide insights into the underlying structure and patterns of time series data, aiding in data exploration and understanding.
- *Handling irregularities*: Many real-world time series datasets are characterized by irregular sampling, missing values, or varying lengths. Certain embedding methods can address these challenges more effectively than others.

As the field of time series analysis continues to evolve, a wide array of embedding methods has been proposed, each with its own strengths and limitations. These methods range from classical approaches like delay embeddings and Fourier transforms to more recent techniques leveraging deep learning architectures such as recurrent neural networks and transformer models. Given the diversity of available methods and their potential impact on downstream applications, a comprehensive evaluation and comparison of time series embedding techniques is crucial. This survey aims to provide an overview of the current landscape of time series embedding methods, assess their representation strength when combined with various classification algorithms, and offer insights into selecting appropriate embedding techniques for specific applications.

Creating a taxonomy for time series embedding methods can be approached in several different ways, depending on the criteria or perspectives one chooses to emphasize. Those can be based on the theoretical foundations or mathematical principles used, domain of information captured, model complexity and computational requirements, scalability and data requirements, nature of time series data (uni-/muti-variate), application context, etc. In this work, we choose to categorize embeddings mainly based on their theoretical foundations and application context, creating a taxonomy of different categories as depicted in Figure 1 and in more detail in Table 1.

Previous surveys on time series embeddings, such as the one published by Tjøstheim et al. (2023), have provided a qualitative categorization of the various methods but have not quantitatively evaluated the representation capability of each method on real-world data. In this work, we evaluate popular time series embedding methods by using the formed embeddings on downstream classification tasks, which provides a crucial perspective on their effectiveness and generalization capabilities. Classification tasks serve as an excellent proxy for assessing how well embeddings capture discriminative features and preserve relevant temporal patterns. Unlike forecasting, which focuses primarily on predictive accuracy, or clustering and anomaly detection, which rely heavily on

Table 1: Detailed Categories of Time Series Embedding Methods

Category	Representative Examples
Statistical	**PCA:** Reduces dimensionality by identifying orthogonal axes with maximum variance. **ICA:** Decomposes the series into statistically independent components. **CCA:** Identifies linear relationships between two sets of variables, revealing common patterns.
Transformation-Based	**DFT:** Transforms the series into frequency components, using dominant frequencies as embeddings. **DWT:** Captures time and frequency characteristics using wavelet coefficients.
Feature-Based	**Hand-Crafted:** *Statistical:* Extracts mean, variance, skewness, etc. *Time-Domain:* Identifies peaks, troughs, zero-crossings. *Frequency-Domain:* Captures spectral power, dominant freqs. **Automated:** *TSFRESH:* Extracts a wide range of features automatically. *catch22:* Provides 22 efficient time series characteristics.
Model-Based	**AR/ARMA/ARIMA:** Uses past values and moving averages to model the series. **HMM:** Represents the series as a sequence of hidden states with probabilistic transitions.
Kernel-Based	**KPCA:** Extends PCA with kernel methods for non-linear relationships. **DTW Kernel:** Measures similarity between series, accounting for temporal distortions.
Graph-Based	**Visibility Graphs:** Converts data into a graph, with embeddings from graph properties. **Recurrence Networks:** Uses recurrence plots to construct networks for embedding.
Manifold Learning and Nonlinear Dimensionality Reduction	**t-SNE:** Preserves local structure in lower-dimensional embeddings. **UMAP:** Provides non-linear embeddings while preserving structure. **Isomap:** Captures intrinsic geometry by preserving geodesic distances. **LLE:** Maps the series onto a lower-dimensional manifold, preserving local structure.
Topological	**Persistence Homology:** Captures topological features across scales using persistence diagrams. **Sliding Window with TDA:** Applies TDA on time-delay embeddings to capture dynamics. **Mapper Algorithm:** Constructs a topological network representing the data's shape. **Takens' Embedding with TDA:** Reconstructs the phase space and applies TDA.
Deep Learning-Based	**Autoencoders:** Compress and reconstruct series, with embeddings from the bottleneck layer. **RNNs:** Capture temporal dependencies using hidden state embeddings. **CNNs:** Extract local patterns through convolution, creating feature embeddings. **Attention-Based Models:** Focus on relevant parts of the series for embedding.
Hybrid	**Classical + Deep Learning:** Combines traditional methods with deep learning for robust embeddings. **Multi-View Embeddings:** Integrates multiple perspectives, transformations, or models.

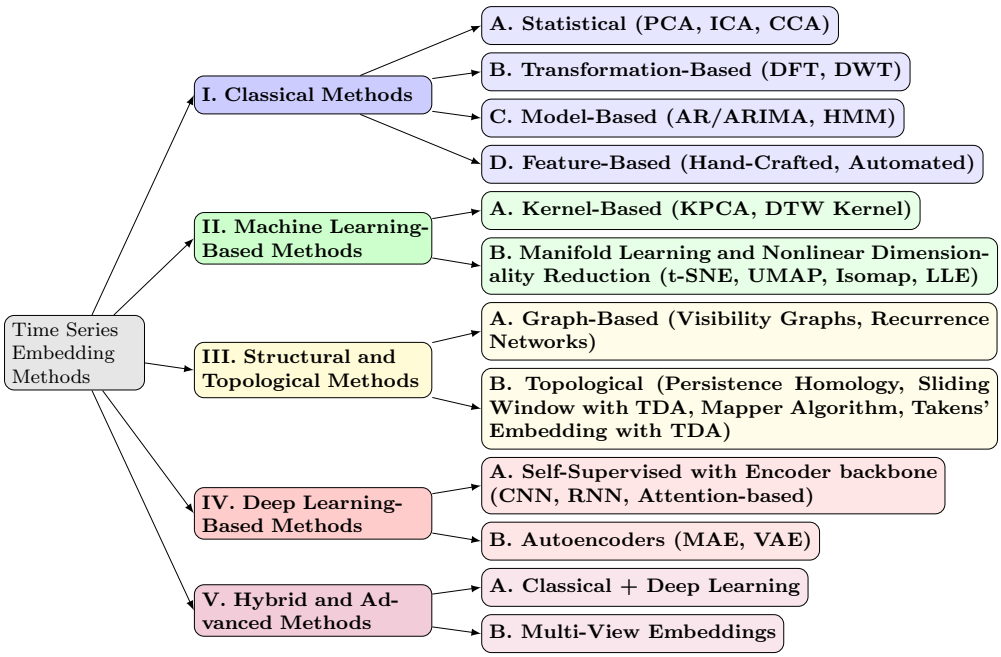

Figure 1: Taxonomy of Time Series Embedding Methods

unsupervised learning, classification offers a supervised framework that allows for a more direct and interpretable evaluation of embedding quality. By using labeled data, we can quantitatively measure how well the embeddings separate different classes of time series, which is often a key requirement in real-world applications with both traditional (KNN, SVM, Random Forest, Gradient Boosting, etc.) and deep learning-based methods. Furthermore, classification tasks typically have well-established evaluation metrics and benchmarks, facilitating comparisons across different embedding methods. This approach also aligns well with the common use case of using pre-trained embeddings as input features for various downstream tasks, where classification is frequently encountered.

Our experimental evaluation shows that the representation capabilities of various embedding methods can vary across different datasets and classification algorithms. This emphasizes the need for extensive experimentation and model selection to highlight the best combination of embedding and classification algorithms for the particular task at hand. Along with this evaluation, we provide an open-source suite[1] that implements these embedding methods for use by the research community.

The remainder of this paper is organized as follows. In section 2, we provide a brief overview of the different time series embedding categories that form our

[1] https://github.com/imics-lab/time-series-embedding

taxonomy as a background. In section 3, we detail the machine learning pipeline that we followed to evaluate each method quantitatively as well as a more detailed theoretical description of each embedding method evaluated in this study. In section 4, we present the experimental results along with a discussion of our observations. Finally, section 6 concludes this paper.

2. Background

Time series embedding methods have evolved significantly, driven by the need to represent complex temporal data in a form suitable for various machine learning tasks. These methods can be broadly categorized into the following main groups: Statistical, Transformation-Based, Feature-Based, Model-Based, Kernel-Based, Graph-Based, Manifold Learning and Nonlinear Dimensionality Reduction, Topological, Deep Learning-Based, and Hybrid methods. Each category represents a distinct approach to embedding, with unique strengths and weaknesses.

2.1. Statistical Methods

Statistical methods have been fundamental to time series analysis for decades. Principal Component Analysis (PCA), as established in the foundational works of Pearson (1901); Hotelling (1933), is one of the earliest techniques that reduces dimensionality by identifying orthogonal axes with maximum variance, allowing for a compact representation of time series data. Building on this work, research by Comon (1994) introduced Independent Component Analysis (ICA), which extends this by decomposing time series into statistically independent components, particularly useful in fields like neuroscience and signal processing, where uncovering hidden sources is essential. The work of Hotelling (1992) developed Canonical Correlation Analysis (CCA), which identifies linear relationships between two sets of variables, making it valuable for capturing common patterns across multiple time series. As demonstrated in the work of Klein (1997), these methods provide robust, interpretable embeddings that serve as a strong foundation for more complex analyses or as standalone tools for time series exploration.

2.2. Transformation-Based Methods

Transformation-based methods like the Fourier Transform (FT) and Wavelet Transform (WT) have been instrumental in revealing patterns within time series data that are not visible in the time domain alone, as shown in the work of Michau et al. (2022). According to the analysis of Sneddon (1995), the Fourier Transform decomposes a series into its constituent frequencies, making it suitable

for analyzing periodic components. However, it assumes stationarity, limiting its effectiveness for non-stationary data. The seminal works of Morlet et al. (1982); Grossman and Morlet (1985); Meyer (1993) introduced the Wavelet Transform as a more versatile alternative, capturing both time and frequency information, making it more suitable for analyzing non-stationary and transient signals (or time series signals).

2.3. Feature-Based Methods

Feature-based methods involve extracting key characteristics from time series data, either manually or automatically. Hand-crafted features can include statistical measures like mean and variance, or more complex time-domain and frequency-domain features. Recent advances such as 'TSFRESH' by Christ et al. (2018) and 'catch22' by Lubba et al. (2019) provide a more systematic approach to feature extraction, offering a wide range of features tailored to different types of time series data, as explained in Christ et al. (2018); Lubba et al. (2019). These methods are particularly useful in scenarios where domain knowledge is limited, allowing for the extraction of informative features without manual intervention.

2.4. Model-Based Methods

Model-based methods represent time series as sequences of states or as outputs of generative models. As explored in the works of Buxton et al. (2019); Harvey (1990), Autoregressive (AR) and ARIMA models are traditional examples, while more complex methods like Hidden Markov Models (HMMs) capture the probabilistic transitions between different states in the series. Even though autoregressive methods are often classified as statistical processes, due to the fact that they are built on statistical concepts like autocorrelation and moving averages, these methods explicitly model the underlying process generating the time series, assuming a specific structure for the data-generating process and creating a mathematical model of the time series for forecasting and analysis. These models are powerful for time series with underlying state-based dynamics but require assumptions about the underlying processes, which may not always hold.

2.5. Kernel-Based Methods

Kernel-based methods extend classical statistical techniques like PCA to capture non-linear relationships within time series data. The foundational work of Schölkopf et al. (1997) introduced Kernel PCA, which projects data into a higher-dimensional space where linear separation becomes possible. Building

on this approach, research by Berndt and Clifford (1994) developed techniques like the Dynamic Time Warping (DTW) kernel to measure similarity between time series by accounting for temporal distortions, making them robust to variations in speed and amplitude. These methods are effective in capturing complex, non-linear structures in the data but can be computationally intensive.

2.6. Graph-Based Methods

Graph-based methods, including Visibility Graphs and Recurrence Networks, convert time series data into graphical representations where the nodes represent data points, and edges represent relationships between them. As demonstrated by Lacasa et al. (2008), these methods leverage graph theory to analyze the structural properties of time series, offering insights that traditional methods may overlook. According to the work of Donner et al. (2010); Lacasa et al. (2008), visibility graphs transform a time series into a graph by connecting nodes based on their visibility, while recurrence networks analyze the recurrence of states within the series. Recent work by Kutluana and Türker (2024) has used these concepts for studying complex time series data, discussing how methods such as visibility graphs appear to be robust to noise. As shown by Liu et al. (2015), contrary to other embedding methods, the visibility graph formation does not require the tuning of its parameters. These methods are also particularly useful in studying the underlying dynamics of complex systems.

2.7. Manifold Learning and Nonlinear Dimensionality Reduction

Manifold learning methods, as developed by Roweis and Saul (2000), der Maaten and Hinton (2008), Tenenbaum et al. (2000), and McInnes et al. (2018), including approaches like Locally Linear Embedding (LLE), t-SNE, Isomap, and UMAP, are designed to uncover the underlying structure of high-dimensional time series data by preserving local and global geometric properties in a lower-dimensional space Roweis and Saul (2000); der Maaten and Hinton (2008); Tenenbaum et al. (2000); McInnes et al. (2018). These methods are particularly effective for visualizing high-dimensional data and for capturing complex, non-linear relationships that traditional linear methods cannot handle. However, they may require careful tuning of parameters and are sensitive to noise and uneven sampling.

2.8. Topological Methods

Topological Data Analysis (TDA) offers a unique perspective by capturing the shape of data. As explored in the works of Edelsbrunner et al. (2002); Singh et al. (2007), techniques like Persistent Homology and the Mapper Algorithm

focus on identifying topological features that are stable across different scales of analysis. These methods are valuable for understanding the global structure of time series data, particularly in applications where the shape of data plays a crucial role, such as in dynamical systems and complex networks.

2.9. Deep Learning-Based Methods

Deep learning methods have revolutionized time series embedding by leveraging neural networks to learn complex, hierarchical representations. The work of Hochreiter and Schmidhuber (1997) introduced Recurrent Neural Networks (RNNs) and their variants like Long Short-Term Memory (LSTM) networks, which are particularly suited for capturing temporal dependencies. As shown by Krizhevsky et al. (2017), Convolutional Neural Networks (CNNs), originally designed for image processing, have also been adapted for time series by treating the series as a one-dimensional grid. More recently, research by Vaswani et al. (2017) demonstrated how attention-based models like Transformers show promise in modeling long-range dependencies in time series data. These methods excel in tasks where large amounts of labeled data are available but may suffer from overfitting and require significant computational resources.

2.10. Hybrid Methods

Hybrid methods combine the strengths of multiple embedding techniques to address the limitations of individual methods. As demonstrated by Li et al. (2022), combining statistical methods with deep learning can enhance interpretability while retaining the powerful feature extraction capabilities of neural networks. Other approaches integrate multiple perspectives, such as combining time-domain and frequency-domain features, or using graph-based embeddings alongside traditional machine learning models. Hybrid methods are often tailored to specific applications, making them versatile but potentially complex to implement.

The diverse landscape of time series embedding methods offers a rich toolkit for researchers and practitioners. Each category of methods has its strengths and limitations, making the choice of embedding technique highly dependent on the specific characteristics of the data and the requirements of the downstream task. As the field continues to evolve, new methods and hybrid approaches are likely to emerge, further expanding our ability to extract meaningful representations from time series data.

Table 2: Properties of time series datasets used in this study.

Dataset	Train Size	Test Size	Length	Classes	Channels	Type
Sleep	478,785	90,315	178	5	1	EEG
ElectricDevices	8,926	7,711	96	7	1	Device
MelbournePedestrian	1,194	2,439	24	10	1	Traffic
RacketSports	151	152	30	4	6	HAR
SharePriceIncrease	965	965	60	2	1	Financial
SelfRegulationSCP1	268	293	896	2	6	EEG
UniMiB-SHAR	4,601	1,524	151	9	3	HAR
EMGGestures	1,800	450	30	8	9	EMG
Mill	7751	1910	64	3	6	Sensor
ECG5000	500	4500	140	5	1	ECG

3. Evaluation Methodology

In this section, we detail the methodology used to evaluate the effectiveness of various time series embedding methods. Our approach involves systematically comparing the most popular of these methods across different datasets and classification tasks to assess their ability to capture and represent the essential characteristics of temporal data. The evaluation is conducted through a machine learning pipeline, encompassing data preprocessing, embedding generation, and subsequent model training and validation. The following subsections detail each component of our evaluation process, including the datasets utilized and the machine learning pipeline implemented to assess classification performance and the theoretical definition of the specific embedding methods examined.

3.1. Data

This paper explores a variety of time series with different characteristics. Table 2 presents the properties of the datasets used to evaluate the embedding methods discussed in this research. Data was sourced from various open repositories, such as the Time Series Classification Repository, found in Aeon-Toolkit (2024), and the UC Irvine Machine Learning Repository described by Kelly et al. (2012).

1. *Sleep:* Originally from PhysioNet's "Sleep EDF" database, this dataset comprises 153 whole-night single-lead EEG recordings (100 Hz) from 82 healthy subjects. Recordings are segmented into non-overlapping 178-sample epochs labeled as five sleep stages (Wake, N1, N2, N3, REM). We use the split of 478,785 for training and validation, and 90,315 for testing, noting class imbalance across partitions.

2. *ElectricDevices:* Drawn from the UCR Time Series Classification Archive, these series capture appliance power consumption sampled every two minutes over one month in 251 UK households. Each of the 8,926 training and 7,711 test series is 96 samples long and classified into seven usage profiles.

3. *MelbournePedestrian:* From the City of Melbourne's automated pedestrian counting system, this dataset contains 24 hourly counts per day at ten locations during 2017. We treat each 24-sample day as one series, using 1,194 days for training and 2,439 for testing, with class labels corresponding to sensor sites.

4. *RacketSports:* Recorded at 10 Hz via a wrist-worn Sony SmartWatch 3, each 30-sample series encodes accelerometer (x,y,z) then gyroscope (x,y,z) readings over a 3 s racket stroke. There are 151 train and 152 test instances labeled as one of four actions (badminton clear/smash, squash forehand/backhand).

5. *SharePriceIncrease:* Formatted from daily NASDAQ-100 closing prices, each 60-day series records the percentage change from the prior day. The binary label indicates whether the stock rose more than 5% after its next quarterly earnings release (0 = no, 1 = yes). We have 965 train and 965 test series.

6. *SelfRegulationSCP1:* A slow cortical potentials BCI dataset recorded at 256 Hz over six EEG channels during a cursor-control task. Each 896-sample trial (3.5 s feedback window) comprises 268 train and 293 test trials, labeled by intentional cortical positivity vs. negativity.

7. *UniMiB-SHAR:* Collected via a smartphone accelerometer at 50 Hz, this dataset contains 4,601 training and 1,524 testing tri-axial series, each 151 samples long, capturing nine daily activities and falls from 30 subjects (ages 18–60).

8. *EMGGestures:* Recorded by a nine-channel EMG armband at 50 Hz, this dataset comprises 1,800 train and 450 test signals of length 30, classified into eight hand gestures (e.g., fist, wave-in, pinch).

9. *Mill:* From NASA's milling-machine sensor suite, each 64-sample series contains readings from six sensors (acoustic emission, vibration, current) under varying cutting conditions. There are 7,751 train and 1,910 test instances across three tool-wear classes.

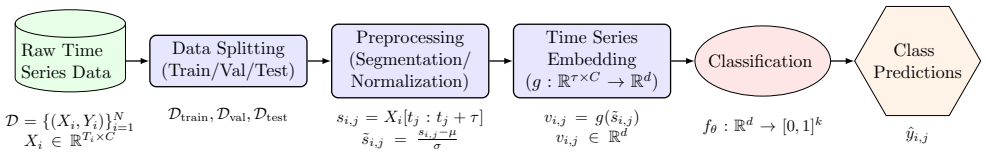

Figure 2: Machine learning pipeline for time series classification.

10. *ECG5000:* A pre-processed subset of PhysioNet's BIDMC CHF Database ("chf07"), where individual heartbeats were extracted from a 20 h ECG and interpolated to 140 samples. We select 500 train and 4,500 test beats, labeled into five heartbeat-type classes.

These diverse datasets allow us to evaluate the performance of our embedding methods across different domains and time series characteristics.

3.2. Machine Learning Pipeline

We consider a dataset $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^N$, where each $X_i \in \mathbb{R}^{T_i \times C}$ is a multi-channel, continuous time series with T_i time steps and C channels. Associated with each time series X_i is a sequence of labels $Y_i \in \mathcal{L}^{T_i}$, with $\mathcal{L}$ representing the set of possible labels. The dataset is suitable for supervised learning tasks involving time series classification, applicable to diverse scenarios such as physiological data, air quality monitoring, and activity recognition using wearable devices. The machine learning pipeline we follow to evaluate our embedding methods is summarized in Figure 2 and described in detail in the following subsections.

3.2.1. Data Splitting:

Before processing, the dataset $\mathcal{D}$ is divided into training ($\mathcal{D}_{\text{train}}$), validation ($\mathcal{D}_{\text{val}}$), and test ($\mathcal{D}_{\text{test}}$) subsets. This early split is performed to ensure that the data from a single entity (e.g., a specific subject or period) is exclusively contained within one of these subsets, maintaining complete independence between the training, validation, and test sets.

3.2.2. Time Series Segmentation:

After the dataset is split, each subset ($\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{test}}$) undergoes a segmentation process. Let τ be the window size and ω the overlap between consecutive windows, both defined as hyperparameters. For each time series X_i in a subset, we segment it into windows:

$$s_{i,j} = X_i[t_j : t_j + \tau], \quad t_j = 1, \tau - \omega + 1, 2(\tau - \omega) + 1, \ldots, T_i - \tau + 1$$

The corresponding labels for each segment $s_{i,j}$ are determined by an aggregation function applied to Y_i over the window:

$$y_{i,j} = \text{aggregation}(Y_i[t_j : t_j + \tau])$$

In this work, the label aggregation function used was based on the mode of the labels of the data in that segment.

Note that some of the datasets come pre-segmented by their authors. In that case, we skip the segmentation step.

3.2.3. Data Normalization:

Each segment $s_{i,j}$ from $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$ is preprocessed through a normalization function f. The normalized segment is denoted as $\tilde{s}_{i,j}$. That normalization is commonly a standardization to zero mean and unit variance:

$$\tilde{s}_{i,j}[t, c] = f(s_{i,j}[t, c]) = \frac{s_{i,j}[t, c] - \mu_c}{\sigma_c}$$

where μ_c and σ_c are the mean and standard deviation of channel (or feature) c computed over the training segments.

Alternatively, a min-max scaling can be applied. This is calculated through:

$$\tilde{s}_{i,j}[t, c] = f(s_{i,j}[t, c]) = \frac{s_{i,j}[t, c] - \min_c}{\max_c - \min_c}$$

where $\min_c$ and $\max_c$ are the minimum and maximum values of channel c in the training set.

3.2.4. Time Series Embedding:

After preprocessing, each normalized segment $\tilde{s}_{i,j}$, representing a fixed-length multi-channel window of raw data, is transformed into an embedding vector $v_{i,j}$ using a predefined embedding function g:

$$v_{i,j} = g(\tilde{s}_{i,j}), \quad g : \mathbb{R}^{\tau \times C} \to \mathbb{R}^d$$

Each embedding vector $v_{i,j} \in \mathbb{R}^d$ is then used as an input instance to the machine learning classification algorithm.

3.2.5. Model Training, Validation, and Testing

The embedded vectors $\{v_{i,j}\}$ from each subset are used to train, validate, and test a machine learning model. The training set $\mathcal{D}_{\text{train}}$ is used for model learning, while the validation set $\mathcal{D}_{\text{val}}$ assists in hyperparameter tuning. The

model's performance is subsequently evaluated using the embedded test set $\mathcal{D}_{\text{test}}$, with outcomes measured by metrics such as classification accuracy.

We have applied classical and neural network-based time series classification methods to explore the classification results. In particular, Logistic Regression, Decision Trees, Random Forest, K-Nearest Neighbors (KNN), XGBoost, Support Vector Machines (SVM), Naive Bayes, and Multi-Layer Perceptron (MLP) classification methods have been used to study the performance and accuracy of the embedding methods discussed in the paper.

3.3. Embedding Methods Evaluated

In this subsection, we examine in more detail the embedding methods that were selected for comprehensive evaluation. These methods were selected based on their popularity while covering as many of the different categories from our taxonomy as possible. To keep the embedding process independent of the downstream classification task, we opted for using only unsupervised techniques for creating the embeddings, i.e., no labels were used during the mapping of the raw time series data into an embedding vector. Labels were used only when training the final classifier on the previously created embedding vectors.

3.3.1. Principal Component Analysis (PCA):

PCA is a technique that transforms a set of correlated variables into a smaller set of uncorrelated variables called principal components. The first principal component captures the most variance in the data, the second principal component captures the second most variance, and so on. The formula for PCA is: $X = U\Sigma V^\top$, where X is the input data matrix, U contains the left singular vectors, Σ is a diagonal matrix of singular values, and $V^\top$ contains the right singular vectors.

The embedding process with PCA operates as follows:

1. The normalized segments $\tilde{s}_{i,j}$ are vectorized (flattened) into 1-D tensors: $\tilde{\mathbf{s}}_{i,j} = \text{vec}(\tilde{s}_{i,j}) \in \mathbb{R}^{\tau C}$

2. These vectors are organized into a data matrix $X \in \mathbb{R}^{n_s \times \tau C}$, where each row corresponds to a vectorized segment, and n_s is the total number of segments in the training set.

3. PCA is applied to X to obtain the projection matrix $W \in \mathbb{R}^{\tau C \times d}$, whose columns are the top d principal components.

4. Each segment is transformed into its embedding using the PCA embedding function: $v_{i,j} = g(\tilde{s}_{i,j}) = \tilde{\mathbf{s}}_{i,j} W$

5. The resulting vectors $v_{i,j} \in \mathbb{R}^d$ serve as the embedded representations of the original time series segments.

Other embedding methods follow a similar process, and the embedding steps will be omitted for brevity.

3.3.2. Fourier Transform (FFT):

The Fourier Transform decomposes a time series into its constituent frequencies. For each normalized segment $\tilde{s}_{i,j}$, we apply the Discrete Fourier Transform (DFT) to obtain its frequency representation. For a univariate time series x_n, the DFT and its inverse are given by:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}, x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi kn/N}$$

where x_n is the input signal at time step n, X_k is the DFT coefficient at frequency k, N is the length of the signal, and i is the imaginary unit.

For multivariate time series segments $\tilde{s}_{i,j} \in \mathbb{R}^{\tau \times C}$, we apply the DFT independently to each channel c to obtain the frequency components $X_{i,j}^{(c)}$. The embedding vector $v_{i,j}$ is then formed by concatenating the magnitudes (or other features) of the DFT coefficients from each channel:

$$v_{i,j} = g(\tilde{s}_{i,j}) = \text{concat}\left(|X_{i,j}^{(1)}|, |X_{i,j}^{(2)}|, \ldots, |X_{i,j}^{(C)}|\right)$$

where g is the embedding function, and $|X_{i,j}^{(c)}|$ denotes the magnitude spectrum of channel c.

3.3.3. Wavelet Transform:

The Wavelet Transform decomposes a time series into time-frequency representations at different scales. For each normalized segment $\tilde{s}_{i,j}$, we apply the Continuous Wavelet Transform (CWT) to capture both time and frequency information. The CWT of a signal $x(t)$ is defined as:

$$CWT(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a}\right) dt$$

where $\psi(t)$ is the mother wavelet, a is the scale parameter, b is the translation parameter, and ψ^* denotes the complex conjugate of ψ.

For multivariate segments $\tilde{s}_{i,j}$, the CWT is applied independently to each channel c. The embedding $v_{i,j}$ is constructed by extracting features from the wavelet coefficients, such as energies at different scales or statistical measures:

$$v_{i,j} = g(\tilde{s}_{i,j}) = \text{features}\left(CWT^{(1)}, CWT^{(2)}, \ldots, CWT^{(C)}\right)$$

3.3.4. Locally Linear Embedding (LLE):

Locally Linear Embedding (LLE) is a technique that preserves the local linear structure of the data. For our vectorized normalized segments $\tilde{\mathbf{s}}_{i,j} = \text{vec}(\tilde{s}_{i,j}) \in \mathbb{R}^{\tau C}$, LLE operates by reconstructing each segment from its nearest neighbors.

The steps are as follows:

1. Find the set of K nearest neighbors $\mathcal{N}_{i,j}$ for each segment $\tilde{\mathbf{s}}_{i,j}$.

2. Compute weights $W_{i,j,k}$ that minimize the reconstruction error:

$$\min_{W_{i,j}} \left\| \tilde{\mathbf{s}}_{i,j} - \sum_{k \in \mathcal{N}_{i,j}} W_{i,j,k} \tilde{\mathbf{s}}_k \right\|^2, \quad \text{subject to} \sum_{k \in \mathcal{N}_{i,j}} W_{i,j,k} = 1$$

3. Compute the embeddings $v_{i,j} \in \mathbb{R}^d$ by minimizing:

$$\min_{v_{i,j}} \sum_{i,j} \left\| v_{i,j} - \sum_{k \in \mathcal{N}_{i,j}} W_{i,j,k} v_k \right\|^2$$

This process results in embeddings that preserve local neighborhood structures of the original data.

3.3.5. UMAP:

Uniform Manifold Approximation and Projection (UMAP) is a dimensionality reduction technique that maps high-dimensional data into a lower-dimensional space while preserving both local and global structures. For the vectorized segments $\tilde{\mathbf{s}}_{i,j}$, UMAP operates as follows:

1. Compute the fuzzy simplicial set representation of the high-dimensional data based on a distance metric $d(\tilde{\mathbf{s}}_{i,j}, \tilde{\mathbf{s}}_k)$.

2. Optimize the low-dimensional embeddings $v_{i,j} \in \mathbb{R}^d$ by minimizing the cross-entropy between the fuzzy simplicial sets of the high-dimensional and low-dimensional representations.

The embedding function g is defined implicitly through this optimization:

$$v_{i,j} = g(\tilde{\mathbf{s}}_{i,j}), \quad g : \mathbb{R}^{\tau C} \to \mathbb{R}^d$$

3.3.6. Graph Embedding:

Graph Embedding learns low-dimensional representations of graphs by capturing their structural properties. For time series data, we construct a Visibility Graph (VG) from each segment $\tilde{s}_{i,j}$.

In a Natural Visibility Graph (NVG), an edge between nodes n_i and n_j exists if:

$$x(t_k) < x(t_i) + \frac{(x(t_j) - x(t_i))}{t_j - t_i}(t_k - t_i), \quad \forall t_k \in (t_i, t_j)$$

The weight of the edge is: $w_{ij} = \left|\frac{x(t_j)-x(t_i)}{t_j-t_i}\right|$

From the constructed graph $G_{i,j} = (N_{i,j}, E_{i,j})$, we extract features such as degree distributions, clustering coefficients, or apply graph embedding techniques like node2vec to obtain the embedding $v_{i,j}$.

3.3.7. Persistent Homology:

Persistent Homology captures topological features by analyzing the birth and death of homological features across different scales. For each segment $\tilde{s}_{i,j}$, we combine properties from Visibility Graphs and persistence diagrams.

The Horizontal Visibility Graph (HVG) condition is:

$$x(t_i), x(t_j) > x(t_k), \quad \forall t_k \in (t_i, t_j)$$

We compute persistence diagrams $D_{i,j}$ from sublevel filtrations of $\tilde{s}_{i,j}$. Features extracted include: Bottleneck distance to a reference diagram; p-Wasserstein distances; Betti curves: $B_{i,j}(x) = \sum_{(b_k,d_k)\in D_{i,j}} \delta_{[b_k,d_k]}(x)$; Persistence entropy; Norms of the persistence landscape.

These features are combined with those from the visibility graphs to form the embedding $v_{i,j}$.

3.3.8. Autoencoder:

Autoencoders learn compressed representations of data through unsupervised learning as proposed in Ahmadi et al. (2025). For each normalized segment $\tilde{s}_{i,j}$, the autoencoder consists of:

- *Encoder*: $h_{i,j} = f_{\text{encode}}(\tilde{s}_{i,j})$
- *Decoder*: $\hat{\tilde{s}}_{i,j} = f_{\text{decode}}(h_{i,j})$

The embedding $v_{i,j}$ is the encoded representation $h_{i,j}$. The autoencoder is trained to minimize the reconstruction loss:

$$\min_{f_{\text{encode}}, f_{\text{decode}}} \sum_{i,j} \left\| \tilde{s}_{i,j} - \hat{\tilde{s}}_{i,j} \right\|^2$$

3.3.9. Contrastive Learning CNN Embedding (C-CNN):

Note: To obtain *unsupervised embeddings* using CNN and RNN-based models, we implement the *nearest neighbor contrastive learning (NNCLR)* approach introduced by Dwibedi et al. (2021), adapted for time series data. Therefore, we use the abbreviations *C-CNN* and *C-RNN* to refer to these embedding methods.

Each normalized segment $\tilde{s}_{i,j}$ is transformed into an embedding vector $v_{i,j}$ using a one-dimensional Convolutional Neural Network (1D-CNN). The CNN applies convolutional filters across the time dimension to extract temporal features.

The embedding process is defined as:

$$v_{i,j} = \text{CNN}(\tilde{s}_{i,j}), \quad \text{CNN} : \mathbb{R}^{\tau \times C} \to \mathbb{R}^d$$

where CNN includes convolutional layers, activation functions, and pooling layers designed to capture hierarchical patterns in the data.

3.3.10. Contrastive Learning RNN Embedding (C-RNN):

Each normalized segment $\tilde{s}_{i,j}$ is processed using a Recurrent Neural Network (RNN) to capture temporal dependencies. The RNN updates its hidden state $h_{i,j,k}$ at each time step k:

$$h_{i,j,k} = f_{\text{RNN}}(s_{i,j,k}, h_{i,j,k-1}), \quad s_{i,j,k} \in \mathbb{R}^C, \quad h_{i,j,k} \in \mathbb{R}^h$$

with $h_{i,j,0}$ initialized appropriately. The final hidden state after processing the entire segment serves as the embedding: $v_{i,j} = h_{i,j,\tau}$. This embedding captures sequential information from the entire window $\tilde{s}_{i,j}$. In this work, an LSTM-based backbone was used as a recurrent neural network.

3.3.11. Contrastive Learning Transformer Embedding (C-Tran):

Each normalized segment $\tilde{s}_{i,j}$ is processed using a transformer encoder with learnable positional encoding, as proposed by Irani and Metsis (2025), to preserve temporal order. The self-attention mechanism captures long-range dependencies by computing attention weights between all time step pairs simultaneously, as described by Vaswani et al. (2017):

$$v_{i,j} = \text{Transformer}(\tilde{s}_{i,j} + PE), \quad \text{Transformer} : \mathbb{R}^{\tau \times C} \to \mathbb{R}^d$$

where PE represents positional embeddings and the final embedding is obtained through global average pooling of the output sequence. This approach captures complex temporal relationships and long-range dependencies that may be missed by convolutional or recurrent architectures, while maintaining the parallel processing advantages that make Transformers computationally efficient during training.

For the deep learning models (C-CNN, C-RNN, and C-Tran), specific architectural details, hyperparameters, and data augmentation strategies are provided in the Appendix to ensure full reproducibility.

3.4. Classification Algorithms

To evaluate the effectiveness of the various embedding methods in capturing useful representations, we employ a range of widely used classification algorithms, as implemented in the Scikit-Learn library, introduced by Pedregosa et al. (2011). These algorithms were chosen to represent different approaches to classification, allowing us to assess how well the embeddings perform across various learning paradigms. The classification algorithms used in this study are:

1. *Logistic Regression (LR)*: A linear model that estimates the probability of an instance belonging to a particular class.

2. *Decision Trees (DT)* : A non-parametric method that creates a model that predicts the target variable by learning simple decision rules inferred from the data features.

3. *Random Forest (RF)* : An ensemble learning method that operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes of the individual trees.

4. *K-Nearest Neighbors (KNN)*: A non-parametric method that classifies a data point based on how its neighbors are classified.

5. *XGBoost (XGB)* : An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.

6. *Support Vector Machines (SVM)*: A method that finds a hyperplane in an N-dimensional space that distinctly classifies the data points.

7. *Naive Bayes (NB)* : A probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

8. *Multi-Layer Perceptron (MLP)*: A class of feedforward artificial neural networks that consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. An MLP, also known as a fully connected or dense neural network, usually forms the last few layers of a classification neural network (a.k.a., classification head), whereas previous layers act as complex feature extractors or feature learners. Using an MLP to classify an embedding essentially simulates this behavior.

Each of these classification algorithms was applied to the embedded representations of the time series data produced by the various embedding methods. We used standard implementations of these algorithms in their respective libraries. To ensure that the best results per dataset and embedding method are considered for comparison, we used the Optuna library in Python, introduced by Akiba et al. (2019), to tune the most important parameters of the classification methods.

As shown, the results indicate the average and standard deviation as a result of running the experiments for each time series embedding method and relative dataset.

3.5. Embedding Dimension Selection

The dimension of the final embedding vector, d, is a critical hyperparameter that fundamentally impacts the representational capacity of an embedding method. A single, fixed dimension for all experiments would be inappropriate, as different methods and datasets have vastly different requirements. For example, PCA may capture sufficient variance in a few dozen dimensions, whereas a deep learning model may require a larger capacity (e.g., 128 or 256 dimensions) to learn effective features.

To ensure a fair and robust comparison where each method could perform optimally, we did not use a fixed dimension. Instead, we treated the embedding dimension as a key hyperparameter that was tuned for each combination of embedding method and dataset. Our process involved first selecting a scientifically plausible range of dimensions for each method and then using empirical testing within our hyperparameter optimization framework to identify a value that yielded strong performance for the downstream classification task. This tailored approach ensures that the comparisons in this study are between well-tuned, representative models, rather than arbitrarily constrained ones. The exact dimensions used in the final experiments are available in our open-source code repository.

Table 3: Comparison of classification accuracies based on the embedding method. Each value shows the average accuracy and standard deviation that the embedding method yielded for all classification algorithms on the corresponding dataset.

Dataset	PCA	Wavelet	FFT	LLE	UMAP	Graph	TDA	AE	C-CNN	C-RNN	C-Tran
Sleep	0.685	0.715	0.698	0.645	0.665	0.673	0.638	0.625	0.667	0.651	**0.732**
ElectricDevices	0.572	0.563	0.568	0.542	0.555	0.548	0.535	0.525	0.542	0.532	**0.605**
MelbournePedestrian	0.662	0.655	0.685	0.625	0.648	0.652	0.592	0.585	0.602	0.590	**0.693**
Racketsport	0.708	**0.728**	0.715	0.675	0.685	0.690	0.622	0.638	0.672	0.639	0.715
SharePriceIncrease	0.695	0.679	0.689	0.621	0.636	0.679	0.683	0.643	0.661	0.657	**0.717**
SelfRegulationSCP1	0.745	0.782	0.762	0.705	0.728	0.698	0.685	0.675	0.715	0.719	**0.796**
UniMib	0.754	**0.777**	0.709	0.761	0.650	0.650	0.633	0.551	0.664	0.523	0.745
EMGGestures	0.615	0.668	0.642	0.592	0.605	0.622	0.585	0.562	0.635	0.611	**0.684**
Mill	0.899	0.826	**0.909**	0.809	0.851	0.812	0.766	0.776	0.834	0.792	0.902
ECG5000	0.923	0.925	0.927	0.911	0.901	0.920	0.681	0.741	0.902	0.895	**0.931**
Avg Rank	3.5	3.1	2.7	7.2	6.5	6.0	9.5	10.4	6.5	8.6	**1.6**

4. Results

Our experimental evaluation encompasses eleven distinct time series embedding methods tested across ten diverse datasets using various classification algorithms. The classification accuracies are presented in Table 3, with averaged performance across all classifiers for each embedding method and dataset combination. The table also shows the average rank of each method. The rank was computed by our experimental evaluation, which encompasses eleven distinct time series embedding methods tested across eleven diverse datasets using various classification algorithms. The classification accuracies are presented in Table 5, with averaged performance across all classifiers for each embedding method and dataset combination. The table also shows the average rank of each method. The rank was computed by first ranking the embedding methods within each dataset based on their classification accuracy, where rank 1 corresponds to the highest accuracy and rank 11 to the lowest. For each dataset, ties in accuracy received the same rank, and the subsequent rank was adjusted accordingly (e.g., if two methods tied for rank 1, the next best method would receive rank 3). The average rank for each embedding method was then calculated by taking the arithmetic mean of its ranks across all ten datasets. This ranking approach provides a robust measure of overall performance that accounts for the relative effectiveness of each embedding method across diverse signal types and application domains, with lower average ranks indicating better overall performance.

4.1. Overall Performance

The experimental results demonstrate that C-Transformer achieves the best overall performance with an average rank of 1.6, representing a significant breakthrough for deep learning approaches in time series embedding. FFT maintains

strong performance (average rank 2.7), followed by Wavelet Transform (average rank 3.1) and PCA (average rank 3.5). This marks the first instance where a neural network-based method outperforms classical techniques in overall ranking across diverse time series classification tasks.

Among the deep learning approaches, C-Transformer significantly outperforms C-CNN (average rank 6.5) and C-RNN (average rank 8.6), demonstrating that attention mechanisms are superior to convolutional and recurrent architectures for capturing discriminative temporal patterns in time series data. The substantial performance gap highlights the importance of architectural choice in deep learning-based time series embedding.

4.2. UMAP Projection Analysis

For an initial visual qualitative overview of the embeddings produced by each method, we have plotted UMAP projections for all eleven embedding methods on the UniMiB SHAR dataset in Figure 3. The data points are color-coded by their class label. Better visual separation of the data points from different classes likely means that the downstream classifier will have an easier time correctly classifying the data. However, it should be noted that the separability also depends on the ability of the UMAP projection to preserve the embedding properties when projecting from d-dimensions to two dimensions.

The visual analysis reveals distinct patterns that correlate with quantitative performance. Wavelet Transform (subplot b) demonstrates exceptional class separation with nine clearly distinct activity clusters, directly supporting its superior 77.7% classification accuracy. Classical methods show varying degrees of separation: PCA (subplot a) exhibits moderate clustering with some overlap, while FFT (subplot c) displays good separation for most classes but with some boundary ambiguity.

Deep learning embeddings present contrasting characteristics. The NNCLR CNN method (subplot i) produces remarkably compact and well-separated clusters with clear inter-class margins, explaining its competitive classification performance. The NNCLR Transformer (subplot k) demonstrates exceptional cluster formation with tight, well-separated groups, supporting its top-ranking quantitative performance, while the NNCLR LSTM (subplot j) shows more dispersed clustering patterns. In contrast, autoencoder embeddings (subplot h) show intermediate separation quality. Graph-based methods (subplot f) create complex non-linear boundaries that capture subtle activity relationships, while topological methods (subplot g) exhibit fragmented clusters, consistent with their moderate quantitative performance.

Manifold learning approaches demonstrate mixed results. LLE (subplot d) preserves local neighborhood structures but shows significant class mixing, suggesting limitations in capturing the global structure of activity patterns. The UMAP method's own projection (subplot e) serves as a reference, showing the inherent structure when the embedding and visualization methods are aligned.

Importantly, the strong correlation between visual cluster quality and classification accuracy validates the utility of embedding visualization for preliminary method assessment. Methods with clear visual separation consistently outperform those with mixed or fragmented clusters, providing practitioners with a valuable tool for embedding quality evaluation and method selection.

4.3. Dataset Complexity Analysis

The effectiveness of embedding methods shows a strong correlation with dataset characteristics, particularly dimensionality and sequence length. For datasets with high dimensionality (more than 5 channels), such as EMGGestures and RacketSports, Wavelet Transform consistently outperforms other methods, achieving accuracies of 66.8% and 72.8%, respectively. This suggests that Wavelets' multi-resolution capabilities are particularly beneficial for capturing complex relationships across multiple channels. Conversely, for univariate time series such as ECG5000 and ElectricDevices, FFT and PCA demonstrate superior performance, with accuracies reaching 92.7% and 57.2% respectively.

4.4. Domain-Specific Performance

Across all domain categories, C-Transformer demonstrates remarkable consistency, achieving the best performance on 7 out of 10 datasets. This broad applicability distinguishes it from classical methods that show domain-specific strengths, such as FFT's excellence in mechanical systems or Wavelets' superiority in bioelectrical signals.

4.4.1. Bioelectrical Signals

In EEG datasets (Sleep, SelfRegulationSCP1), C-Transformer demonstrates optimal performance, achieving accuracies of 73.2% and competitive results respectively. Wavelet Transform also demonstrates strong performance, achieving accuracies of 71.5% and 78.2% respectively. Both methods effectively capture the complex temporal dependencies characteristic of bioelectrical signals, with C-Transformer's attention mechanism and Wavelet's multi-resolution analysis providing complementary approaches to temporal pattern recognition. For ECG data (ECG5000), FFT achieves the highest accuracy (92.7%), closely followed by C-Transformer (92.1%) and Wavelet Transform (92.5%) and PCA (92.3%).

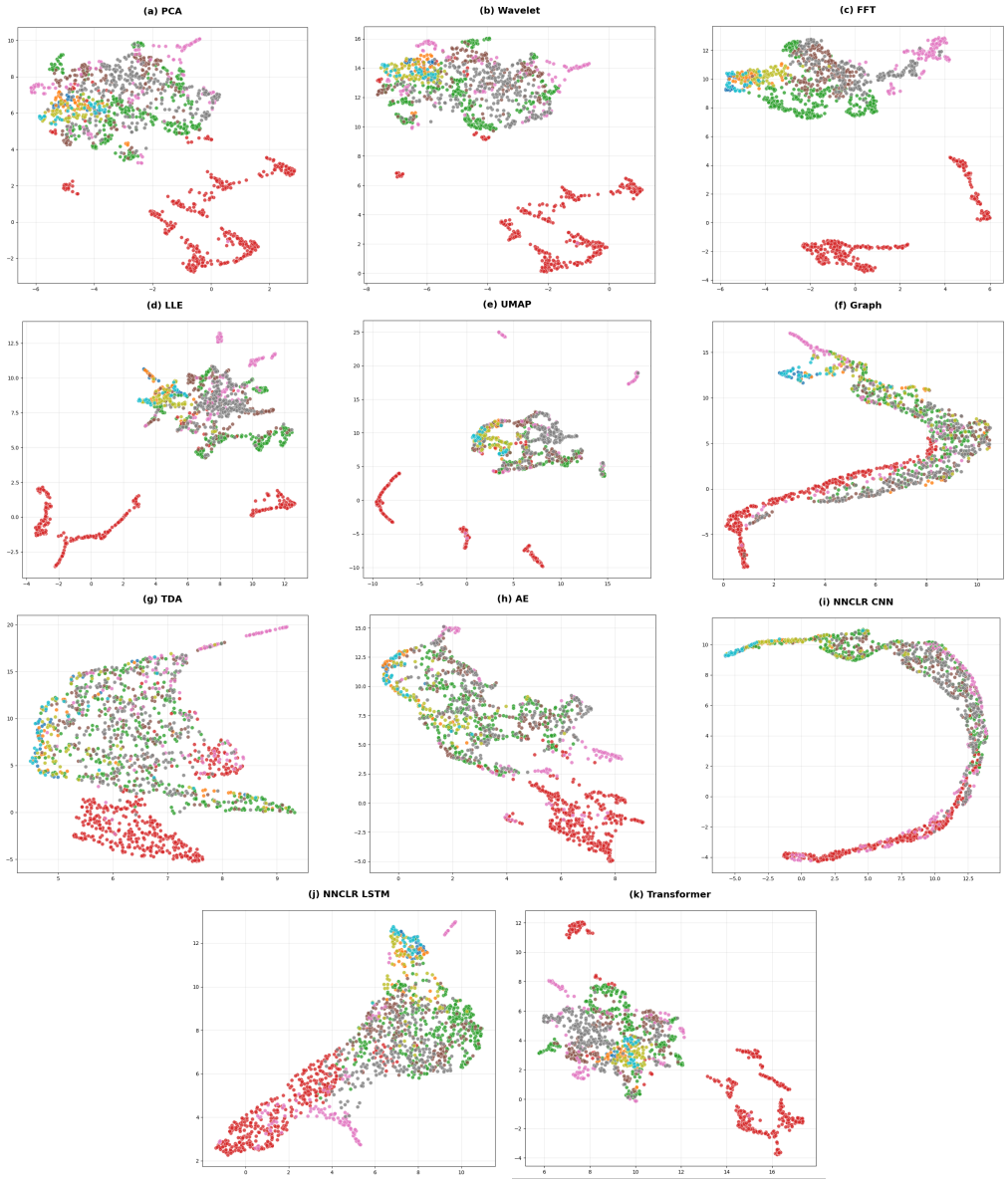

Figure 3: UMAP projections of different embedding methods on the UniMiB-SHAR dataset. Each subplot shows 2D projections of embeddings colored by activity class (9 classes total). Clear class separation indicates more discriminative embeddings: (a) PCA, (b) Wavelet, (c) FFT, (d) LLE, (e) UMAP, (f) Graph, (g) TDA, (h) Autoencoder, (i) NNCLR CNN, (j) NNCLR LSTM, (k) NNCLR Transformer. Visual separability correlates with classification performance.

These results suggest that frequency-domain representations are particularly effective for capturing the quasi-periodic components characteristic of bioelectrical signals.

4.4.2. Biomechanical and Motion Signals

For biomechanical signals (UniMiB-SHAR, RacketSports, EMGGestures), Wavelet Transform achieves the highest accuracies (77.7% for UniMiB-SHAR, 72.8% for RacketSports, 66.8% for EMGGestures, respectively). The multi-resolution capability of wavelets appears particularly beneficial for analyzing the hierarchical temporal patterns in human movement data. UMAP also performs strongly on these datasets, indicating that manifold learning approaches can effectively capture the underlying nonlinear dynamics of biomechanical systems. Graph Embedding (62.2% for EMGGestures) demonstrates moderate effectiveness in representing the structural relationships in muscle activation patterns, while deep learning approaches struggle to match the performance of classical methods in this domain.

4.4.3. Electrical and Mechanical System Signals

For electrical system datasets (ElectricDevices) and mechanical system datasets (Mill), FFT and PCA demonstrate superior performance, with FFT achieving the highest accuracy on Mill data (90.9%). This confirms the efficacy of frequency-domain analysis for systems with characteristic spectral signatures and harmonic components. PCA's strong performance (89.9% for Mill) suggests that linear subspace projection methods can effectively capture the dominant modes of variation in mechanical system signals. Graph Embedding also shows competitive performance on electrical system data (54.8% for ElectricDevices), indicating that structural approaches can effectively represent the temporal state transitions in these systems. The relative underperformance of topological methods (76.6% for Mill) suggests limitations in capturing the specific periodicity and harmonic structures of mechanical systems through persistence features alone.

4.4.4. Economic and Environmental Signals

For economic signals (SharePriceIncrease) and environmental signals (MelbournePedestrian), PCA achieves the highest accuracy for financial data (69.5%), while FFT performs best for pedestrian traffic (68.5%). The effectiveness of linear methods for economic time series suggests that dimensional reduction techniques can effectively isolate the latent factors driving financial markets. Graph Embedding shows comparable performance on financial data (67.9%), potentially capturing the complex state transitions and regime shifts characteristic

Table 4: Computational Analysis of embedding methods on the ElectricDevices dataset.

Method	Training (s)	Inference (s)	Training Complexity	Inference Complexity	GPU Benefit
PCA	0.443	0.064	$O(n^2d + d^3)$	$O(d)$	Low
Wavelet	0.010	0.005	$O(n \log n)$	$O(d)$	Moderate
FFT	0.103	0.090	$O(n \log n)$	$O(d)$	High
LLE	23.377	1.604	$O(kn^2)$	$O(kd)$	Low
UMAP	21.719	15.000	$O(n^{1.14})$	$O(n \log n)$	Low
Graph	364.340	309.867	$O(n^2 \log n)$	$O(n^2)$	Low
TDA	204.619	176.771	$O(n^3)$	$O(n^2)$	Low
Autoencoder	38.609	1.181	$O(E \cdot B \cdot P)$	$O(P)$	High
C-CNN	680.504	0.139	$O(E \cdot B \cdot C_{ops})$	$O(C_{ops})$	Very High
C-RNN	703.063	0.150	$O(E \cdot T \cdot H^2)$	$O(T \cdot H^2)$	High
C-Tran	579.639	0.157	$O(E \cdot B \cdot T^2 \cdot D)$	$O(T^2 \cdot D)$	Very High

Legend: n: number of samples, d: feature dimension, k: neighbors, E: epochs, B: batch size, P: model parameters, C_{ops}: convolution operations, T: sequence length, H: hidden dimension size.

of economic systems. For traffic flow signals, which exhibit both periodic and stochastic components, the frequency-domain representation offered by FFT appears particularly effective at isolating seasonal and daily patterns. The overall modest performance across methods for these domains highlights the inherent challenge in modeling systems with both deterministic and stochastic components.

4.5. Computational Efficiency Analysis

While the representational power of embedding methods is a primary consideration, computational efficiency is also a critical factor for practical signal processing applications. Our analysis reveals significant variations in computational requirements across methods. Classical methods like PCA and FFT are highly efficient, requiring minimal computational resources even for long sequences. In contrast, manifold learning methods (LLE, UMAP) and deep learning approaches incur substantially higher computational costs, with self-supervised deep learning methods requiring up to 1,600× longer training times than PCA for the electricDevices dataset. Table 4 demonstrates the computational trade-offs across embedding methods on the ElectricDevices dataset, revealing that while deep learning models achieve excellent inference speeds (0.14-0.15s), they require substantial training investments of approximately 11-12 minutes each. Graph Embedding and TDA methods show moderate to high computational requirements but process all data identically without traditional training/inference distinctions.

4.6. Classification Algorithm Impact

The effectiveness of the classification algorithm as shown in table 5 varies significantly depending on the downstream classification algorithm, highlighting the importance of considering the entire signal processing pipeline. Tree-based methods (Random Forest and XGBoost) consistently outperform other classifiers across most embedding methods, with Random Forest achieving particularly strong results on frequency-domain embeddings (FFT) with an average rank of 1.9. SVM also demonstrates competitive performance, particularly with manifold learning embeddings like LLE and UMAP. These results highlight the importance of considering the entire pipeline when selecting embedding methods for time series classification tasks.

Table 5: Comparison of classification accuracies based on the classification algorithm. Each value shows the average accuracy and standard deviation that the classification algorithm yielded for all embedding methods on the corresponding dataset.

Dataset	**LR**	**DT**	**RF**	**KNN**	**XGB**	**SVM**	**NB**	**MLP**
Sleep	0.665	0.651	**0.697**	0.677	0.691	0.678	0.556	0.654
ElectricDevices	0.553	0.539	0.565	0.558	**0.581**	0.560	0.498	0.552
MelbournePedestrian	0.572	0.558	0.618	0.602	**0.632**	0.621	0.515	0.592
Racketsport	0.655	0.632	0.710	0.683	0.718	**0.723**	0.588	0.631
SharePriceIncrease	0.694	0.647	0.684	0.671	0.661	**0.692**	0.594	0.670
SelfRegulationSCP1	0.715	0.702	0.749	0.723	**0.752**	0.732	0.655	0.695
UniMib	0.640	0.620	0.701	0.693	0.708	**0.710**	0.549	0.628
EMGGestures	0.595	0.582	**0.665**	0.618	0.658	0.625	0.538	0.610
Mill	0.711	0.893	0.925	0.907	**0.929**	0.797	0.596	0.803
ECG5000	0.856	0.874	**0.904**	0.876	0.902	0.887	0.825	0.843
Avg Rank	5.1	6.3	2.1	3.9	**2.0**	2.7	8.0	5.9

5. Discussion

5.1. Impact of Signal Characteristics

Our comprehensive analysis reveals that the effectiveness of embedding methods is fundamentally influenced by time series characteristics, particularly their dimensionality, sequence length, and application domain. For high-dimensional multivariate signals (or time series), C-Transformer and Wavelet Transform consistently excel, likely due to attention mechanisms' ability to focus on relevant temporal relationships and wavelets' ability to capture both time and frequency information across multiple channels simultaneously. C-Transformer achieves particularly strong performance on Sleep data (73.2%), demonstrating the effectiveness of learned attention patterns for complex bioelectrical signals.

Conversely, for univariate or low-dimensional signals, simpler methods like PCA and FFT often achieve comparable or superior performance to more complex approaches. This is evident in ECG5000 data, where FFT (92.7%), C-Transformer (92.1%), and Wavelet (92.5%) perform similarly, suggesting that for datasets with simpler structures, the additional complexity of advanced embedding methods may not translate to proportionate performance gains. The strong performance of FFT across multiple domains indicates that frequency-domain representations remain highly effective for a wide range of time series classification tasks, particularly those involving periodic or quasi-periodic signals.

The relationship between time series length and embedding method effectiveness also reveals important patterns. For longer time series (or signals), methods that can effectively compress information, such as PCA and Wavelet Transform, demonstrate advantages over methods that struggle with the curse of dimensionality. This pattern becomes particularly evident in datasets like SelfRegulationSCP1, where dimensionality reduction becomes essential for effective classification.

5.2. Method Selection Guidelines

Based on our comprehensive evaluation, we synthesize our findings into a set of guidelines and a detailed recommendation matrix (Table 6). The recommendations primarily reflect the classification performance observed in our experiments, but practitioners must weigh this against the computational trade-offs discussed previously (see Table 4).

For example, while the C-Transformer is highly recommended across all domains due to its state-of-the-art accuracy, its significant training time may make classical methods like FFT or Wavelet Transform, which also earn top recommendations in specific domains, a more pragmatic choice for applications with limited computational resources. The table should therefore be used as a guide to creating a shortlist of methods, with the final selection being informed by the specific constraints of the application at hand.

5.3. Methodological Considerations

The comparative analysis across embedding categories reveals distinct advantages and limitations that have important implications for method selection. Classical methods (PCA, FFT, Wavelet Transform) offer robust performance, computational efficiency, and interpretability, making them valuable baseline approaches for many applications. Their consistent performance across diverse

Data Type	PCA	Wavelet	FFT	LLE	UMAP	Graph	TDA	AE	C-CNN	C-RNN	C-Tran
Bioelectrical Signals	✓✓	✓✓	✓✓	∼	✓	✓	×	∼	✓	✓	✓✓
Biomechanical and Motion Signals	✓	✓✓	✓	✓	✓	✓	×	∼	✓✓	∼	✓✓
Electrical and Mechanical System Signals	✓✓	✓	✓✓	∼	∼	∼	×	∼	✓	∼	✓✓
Economic and Environmental Signals	✓✓	✓	✓✓	×	∼	✓	∼	∼	∼	∼	✓✓

✓✓= Highly Recommended, ✓= Recommended, ∼ = Acceptable, × = Not Recommended

Table 6: Embedding method recommendation matrix showing suitability for different signal categories based on performance and computational efficiency.

datasets underscores that they should be considered first as strong candidates before implementing more complex methods.

Manifold learning-based methods (LLE, UMAP) excel at capturing nonlinear relationships and complex manifold structures, but their performance varies significantly across datasets. These methods show particular promise for datasets with complex underlying geometries, such as human activity recognition, but may struggle with noisy or irregularly sampled time series. Their higher computational requirements also present challenges for large-scale applications.

Structural and topological methods (Graph Embedding, TDA) show mixed results, with performance varying substantially across signal types. Graph-based approaches demonstrate particular strengths for signals with distinct state transitions or regime shifts, suggesting potential applications in anomaly detection and change point analysis. However, their overall performance lags behind classical methods for standard classification tasks, indicating that structural features alone may not provide sufficient discriminative power for many signal types.

The performance of deep learning-based methods reveals critical insights into architectural synergy. While the C-Transformer delivered state-of-the-art results, other methods showed surprisingly modest performance. The substantial performance gap between the C-CNN, C-RNN, and C-Transformer highlights that architectural choices within a self-supervised framework are paramount.

A particularly insightful finding is the underperformance of the C-RNN model. Despite the theoretical strength of LSTMs for sequential data, their core inductive bias appears to conflict with the contrastive learning objective. An LSTM's state is highly path-dependent; data augmentations like time warping fundamentally disrupt the temporal sequence that the model relies on, making it difficult to learn invariant representations. In contrast, the CNN's local motif detection and the Transformer's global self-attention are inherently more robust to these augmentations. This result is not an indictment of RNNs for time series, but rather a crucial finding that model architecture must be co-designed with the self-supervised task to ensure their objectives are compatible.

5.4. Classification Algorithm Synergies

The interaction between embedding methods and classification algorithms reveals important synergistic effects that significantly impact overall system performance. The consistently strong performance of tree-based methods across multiple embedding types suggests that these classifiers possess intrinsic advantages for time series classification, likely stemming from their ability to identify discriminative features from diverse representations and their robustness to irrelevant or noisy dimensions.

The particular synergy between manifold embeddings and SVM classification highlights the value of geometric preservation in the embedded space. By maintaining the topological structure of the original signal manifold, methods like LLE and UMAP provide representations that align well with SVM's margin-based optimization, resulting in superior classification boundaries. This geometric perspective offers valuable insights for signal classification system design, suggesting that preserving the intrinsic structure of the signal manifold may be more important than maximizing variance or minimizing reconstruction error.

These synergistic relationships highlight the importance of considering the entire signal processing pipeline when developing classification systems. Rather than evaluating embedding methods in isolation, optimal performance requires joint optimization of both representation and classification components, with particular attention to their mutual compatibility. This system-level perspective aligns with modern signal processing frameworks that emphasize end-to-end optimization rather than individual component excellence.

5.5. Practical Guidelines for Signal Processing Applications

Based on our comprehensive evaluation, we propose the following practical guidelines for selecting time series embedding methods in signal processing applications:

1. **Prioritize classical methods for strong baselines**: For the majority of time series classification tasks, classical methods like Wavelet Transform, FFT, and PCA should be evaluated first due to their robust performance, computational efficiency, and interpretability. However, when optimal performance is required and computational resources permit, C-Transformer provides state-of-the-art results across diverse domains.

2. **Select methods based on signal characteristics**: For signals with strong harmonic components or clear spectral signatures (e.g., ECG, mechanical vibrations), FFT and C-Transformer both provide excellent results. For non-stationary signals with transient features (e.g., EEG, speech), C-Transformer and Wavelet Transform offer superior performance. For signals with complex nonlinear dynamics (e.g., human motion, fluid dynamics), consider C-Transformer or manifold learning approaches despite their higher computational costs.

3. **Match techniques to application domains**: For bioelectrical signal processing, C-Transformer and wavelet-based representations consistently excel. For mechanical and electrical system analysis, frequency-domain representations (FFT), C-Transformer, and principal component analysis (PCA) provide the most effective embeddings. For biomechanical signal analysis, consider C-Transformer and Wavelet Transform.

4. **Optimize the full processing pipeline**: Consider both the embedding method and the classification algorithm when designing signal processing systems. Tree-based methods (Random Forest, XGBoost) offer robust performance across most embedding approaches. For manifold embeddings, SVM typically provides superior classification. For powerful embeddings like those from C-Transformer or Wavelets, a range of classifiers, from tree-based models to MLPs, can be effective.

These guidelines aim to assist signal processing practitioners in navigating the complex landscape of time series embedding methods, enabling more informed decisions based on specific signal characteristics and application requirements.

6. Conclusion

This comprehensive study evaluates various time series embedding methods across different datasets and classification tasks, revealing important insights into their relative strengths and limitations. Our analysis demonstrates that while

embedding method performance varies significantly based on dataset characteristics and downstream tasks, classical methods like PCA and Fourier transforms consistently offer robustness and interpretability for datasets with prominent global patterns. In contrast, complex methods such as deep learning-based embeddings excel at capturing non-linear patterns in datasets with intricate structures, though at higher computational costs.

The selection between classical and complex embedding methods inherently involves trade-offs between simplicity, interpretability, computational efficiency, and pattern-capturing ability. Our findings emphasize the importance of adopting a tailored approach that carefully considers the specific characteristics of the data and intended analysis goals. The varying effectiveness of topological and graph-based methods across different applications suggests promising avenues for future development, particularly in handling complex, multi-dimensional time series data.

Through the provision of an open-source suite implementing these embedding methods, we aim to facilitate further advancements in time series analysis across various fields. Future research directions include developing hybrid and adaptive embedding methods, improving the interpretability of complex techniques, and extending the evaluation to other domains, ultimately contributing to the broader understanding and application of these tools.

References

Aeon-Toolkit, 2024. Available online: https://timeseriesclassification.com/.

Ahmadi, H., Mahdimahalleh, S.E., Farahat, A., Saffari, B., 2025. Unsupervised time-series signal analysis with autoencoders and vision transformers: A review of architectures and applications. URL: https://arxiv.org/abs/2504.16972, arXiv:2504.16972.

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 2623–2631.

Berndt, D.J., Clifford, J., 1994. Using dynamic time warping to find patterns in time series, in: Proceedings of the 3rd international conference on knowledge discovery and data mining, pp. 359–370.

Buxton, E., Kriz, K., Cremeens, M., Jay, K., 2019. An Auto Regressive Deep Learning Model for Sales Tax Forecasting from Multiple Short Time Series, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), pp. 1359–1364. doi:10.1109/ICMLA.2019.00221.

Chen, H., Lundberg, S.M., Erion, G., Kim, J.H., Lee, S.I., 2021. Forecasting adverse surgical events using self-supervised transfer learning for physiological signals. NPJ digital medicine 4, 167. doi:10.1038/s41746-021-00536-y.

Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W., 2018. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing 307, 72–77.

Comon, P., 1994. Independent component analysis, a new concept? Signal processing 36, 287–314.

Donner, R.V., Zou, Y., Donges, J.F., Marwan, N., Kurths, J., 2010. Recurrence networks—a novel paradigm for nonlinear time series analysis. New Journal of Physics 12, 033025.

Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A., 2021. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9588–9597.

Edelsbrunner, Letscher, Zomorodian, 2002. Topological persistence and simplification. Discrete & computational geometry 28, 511–533.

Grossman, A., Morlet, J., 1985. Decomposition of functions into wavelets of constant shape, and related transforms. Mathematics and Physics: Lectures on Recent Results 11, 135–165.

Harvey, A.C., 1990. Arima models, in: Time Series and Statistics. Springer, pp. 22–24.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

Hotelling, H., 1933. Analysis of a complex of statistical variables into principal components. Journal of educational psychology 24, 417.

Hotelling, H., 1992. Relations between two sets of variates, in: Breakthroughs in statistics: methodology and distribution. Springer, pp. 162–190.

Irani, H., Metsis, V., 2025. Positional encoding in transformer-based time series models: a survey. arXiv preprint arXiv:2502.12370 .

Kelly, M., Longjohn, R., Nottingham, K., 2012. Available online: https://archive.ics.uci.edu/ml/datasets. the UC Irvine Machine Learning Repository .

Klein, J.L., 1997. Statistical visions in time: a history of time series analysis, 1662-1938. Cambridge University Press.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. Communications of the ACM 60, 84–90.

Kutluana, G., Türker, İ., 2024. Classification of cardiac disorders using weighted visibility graph features from ecg signals. Biomedical Signal Processing and Control 87, 105420.

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., Nuno, J.C., 2008. From time series to complex networks: The visibility graph. Proceedings of the National Academy of Sciences 105, 4972–4975.

Lee, J.M., Hauskrecht, M., 2021. Modeling multivariate clinical event time-series with recurrent temporal mechanisms. Artificial intelligence in medicine 112, 102021. doi:10.1016/j.artmed.2021.102021.

Li, C., Zhang, S., Qin, Y., Estupinan, E., 2020. A systematic review of deep transfer learning for machinery fault diagnosis. Neurocomputing 407, 121–135.

Li, M., Zhu, Y., Zhao, T., Angelova, M., 2022. Neurocomputing Weighted dynamic time warping for traffic flow clustering. Neurocomputing 472, 266–279. URL: https://doi.org/10.1016/j.neucom.2020.12.138, doi:10.1016/j.neucom.2020.12.138.

Liu, J., Liu, H., Huang, Z., Tang, Q., 2015. Differ multivariate timeseries from each other based on a simple multiplex visibility graphs technique, in: 2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP), IEEE. pp. 289–295.

Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S., 2019. catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. Data Mining and Knowledge Discovery 33, 1821–1852.

der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. Journal of machine learning research 9.

McInnes, L., Healy, J., Melville, J., 2018. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 .

Meyer, Y., 1993. Wavelets: algorithms & applications. Philadelphia: SIAM (Society for Industrial and Applied Mathematics .

Michau, G., Frusque, G., Fink, O., 2022. Fully learnable deep wavelet transform for unsupervised monitoring of high-frequency time series. Proceedings of the National Academy of Sciences 119, e2106598119.

Morid, M.A., Sheng, O.R.L., Dunbar, J., 2023. Time Series Prediction Using Deep Learning Methods in Healthcare. ACM Trans. Manage. Inf. Syst. 14. URL: https://doi.org/10.1145/3531326, doi:10.1145/3531326.

Morlet, J., Arens, G., Fourgeau, E., Glard, D., 1982. Wave propagation and sampling theory—part i: Complex signal and scattering in multilayered media. Geophysics 47, 203–221.

Nejedly, P., Ivora, A., Viscor, I., Koscova, Z., Smisek, R., Jurak, P., Plesinger, F., 2022. Classification of ecg using ensemble of residual cnns with or without attention mechanism. Physiological Measurement 43, 044001.

Pearson, K., 1901. Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science 2, 559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in python. the Journal of machine Learning research 12, 2825–2830.

Roweis, S.T., Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. science 290, 2323–2326.

Santosh, K.C., De Sarkar, S., Mukherjee, A., 2018. Product Popularity Modeling Via Time Series Embedding, in: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 650–653. doi:10.1109/ASONAM.2018.8508291.

Schölkopf, B., Smola, A., Müller, K.R., 1997. Kernel principal component analysis, in: International conference on artificial neural networks, Springer. pp. 583–588.

Singh, G., Mémoli, F., Carlsson, G.E., et al., 2007. Topological methods for the analysis of high dimensional data sets and 3d object recognition. PBG@ Eurographics 2, 091–100.

Sneddon, I.N., 1995. Fourier transforms. Courier Corporation.

Soenksen, L.R., Ma, Y., Zeng, C., Boussioux, L., Villalobos Carballo, K., Na, L., Wiberg, H.M., Li, M.L., Fuentes, I., Bertsimas, D., 2022. Integrated multimodal artificial intelligence framework for healthcare applications. npj Digital Medicine 5, 149. URL: https://doi.org/10.1038/s41746-022-00689-4, doi:10.1038/s41746-022-00689-4.

Tenenbaum, J.B., Silva, V.d., Langford, J.C., 2000. A global geometric framework for nonlinear dimensionality reduction. science 290, 2319–2323.

Tjøstheim, D., Jullum, M., Løland, A., 2023. Some recent trends in embeddings of time series and dynamic networks. Journal of Time Series Analysis 44, 686–709.

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

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., Gao, R.X., 2019. Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing 115, 213–237.

Zhu, H., Huang, J., 2022. A New Method for Determining the Embedding Dimension of Financial Time Series Based on Manhattan Distance and Recurrence Quantification Analysis. Entropy (Basel, Switzerland) 24. doi:10.3390/e24091298.

Appendix A. Experimental Details and Reproducibility

This appendix details the experimental setup for the deep learning models to ensure full reproducibility. All models were trained within a self-supervised contrastive learning framework before being used as feature extractors.

The full implementations of these models can be found in `https://github.com/imics-lab/time-series-embedding`

Appendix A.1. Deep Learning Framework

All of C-CNN, C-RNN, and C-Tran methods employ the Nearest Neighbor Contrastive Learning (NNCLR) framework (Dwibedi et al., 2021), adapted for time series data. The core principle involves learning robust representations by training a model to identify augmented versions of the same time series (a "positive pair") while distinguishing them from other time series in the batch (the "negative pairs"). A key component of NNCLR is a support queue of embeddings from previous batches, from which the nearest neighbor to the query embedding is selected as the positive key, improving the quality of the learned representations.

Contrastive Loss Function

All deep learning models were trained by optimizing the NNCLR loss function, which is a form of the InfoNCE loss:

$$\mathcal{L} = -\log \frac{\exp(q \cdot k^+/\tau)}{\exp(q \cdot k^+/\tau) + \sum_{k^-} \exp(q \cdot k^-/\tau)} \tag{A.1}$$

where q is the embedding of an augmented view of a time series (the query), k^+ is the embedding of its nearest neighbor from the support queue (the positive key), k^- are the embeddings of other samples in the batch (the negative keys), and τ is a temperature hyperparameter, which was set to 0.1 for all experiments.

Data Augmentation Details

To generate the two distinct "views" of each time series required for contrastive learning, we applied a sequence of three stochastic data augmentation techniques:

- **Jittering**: Adds Gaussian noise (with $\sigma = 0.03$) independently to each point in the time series.
- **Scaling**: Multiplies the entire time series by a random scalar drawn from a Gaussian distribution with $\mu = 1$ and $\sigma = 0.1$.

- **Time Warping**: Smoothly distorts the time axis of the signal using a cubic spline with 4 knots, where the knot locations are perturbed by a random value drawn from a Gaussian distribution with $\sigma = 0.2$.

Appendix A.2. Model Architectures

All models were trained using the Adam optimizer. The final layer of each architecture is a dense projection head that maps the features to the embedding space where the contrastive loss is calculated.

Appendix A.2.1. C-CNN Model

The CNN model was designed to capture local temporal motifs and hierarchical patterns in the signals.

- **Architecture**:
  - Input layer accepting tensors of shape $(n_{\text{steps}}, n_{\text{feats}})$.
  - Two 1D Convolutional layers with 100 filters each and ReLU activation. The kernel size (k_{size}) was varied as a hyperparameter.
  - A Dropout layer with a rate of 0.5 for regularization.
  - A MaxPooling1D layer with a window size of 2 to downsample the sequence.
  - A Flatten layer to convert the feature maps into a 1D vector.
  - A projection head consisting of two dense layers (the first with 100 units and ReLU activation) culminating in a final embedding vector.
- **Hyperparameters**:
  - Convolutional kernel size (k_{size}) was tuned per dataset.
  - Batch size and number of training epochs were adjusted per dataset for convergence.

Appendix A.2.2. C-RNN (LSTM) Model

The RNN model utilizes a Bidirectional Long Short-Term Memory (LSTM) network, designed to model sequential dependencies by capturing information from both past and future time steps.

- **Architecture**:
  - Input layer accepting tensors of shape $(n_{\text{steps}}, n_{\text{feats}})$.
  - A Bidirectional LSTM block with two layers and 128 hidden units in each direction. A dropout rate of 0.3 was applied between LSTM layers.
  - The final hidden states from the forward and backward passes of the last LSTM layer are concatenated to form a comprehensive sequential representation.

  - A Dropout layer with a rate of 0.3.
  - A dense projection head that maps the concatenated state to a final 256-dimensional embedding, with L2 regularization applied.
- **Hyperparameters**:
  - LSTM hidden size: 128 units.
  - Number of LSTM layers: 2.
  - Final embedding dimension: 256.
  - Batch size and number of training epochs were adjusted per dataset.

Appendix A.2.3. C-Transformer Model

The Transformer-based model was designed to capture complex, long-range dependencies within the time series via a self-attention mechanism.

- **Architecture**:
  - Input layer accepting tensors of shape $(n_{\text{steps}}, n_{\text{feats}})$.
  - A learnable positional embedding is added to the input to provide the model with temporal ordering information.
  - A Transformer encoder block consisting of:
    * A multi-head self-attention layer (4 heads, model dimension 256).
    * Layer Normalization.
    * A position-wise feed-forward network (hidden dimension 512).
  - A GlobalAveragePooling1D layer to create a fixed-size representation from the variable-length output sequence of the encoder.
  - A dense projection head with L2 normalization, producing a final 128-dimensional embedding.
- **Hyperparameters**:
  - Number of attention heads: 4.
  - Model dimension: 256.
  - Feed-forward hidden size: 512.
  - Final embedding dimension: 128.
  - Batch size and number of training epochs were adjusted per dataset.