universität wien

MASTERARBEIT / MASTER'S THESIS

Titel / Title

DipContrast: A Contrastive Learning Framework for Non-Parametric Deep Time-Series Clustering

verfasst von / submitted by

Moinuddin Noor BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2026 / Vienna, 2026

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Master's degree programme Computer Science
Betreut von / Supervisor:	Ass.-Prof. Dott.ssa Dott.ssa.mag.Yllka Velaj, PhD

Acknowledgements

I would like to extend my thanks to the entire supervisory team, especially Prof. Yllka Velaj, Lena Bauer, and Pascal Weber. Thank you Gerhard. I would also like to thank my family, friends, and all the people who believe in me.

Abstract

Time-series data is ubiquitous across domains such as finance, healthcare, and energy systems, where uncovering latent patterns is essential for downstream analysis. Clustering time-series data remains challenging due to high dimensionality, temporal dependencies, and the often unknown number of clusters (k). While deep time-series clustering methods have shown promising results by integrating representation learning with clustering, most existing approaches are parametric and rely primarily on time-domain representations.

To address these limitations, we propose **DipContrast**, an end-to-end contrastive learning framework for non-parametric deep time-series clustering. DipContrast leverages a dual-domain architecture to learn robust latent representations from both time and frequency domains. The framework operates without prior knowledge of k by initially overestimating the number of clusters and progressively refining the latent space. It employs a prototype-driven non-parametric mechanism that iteratively merges candidate clusters based on unimodality, as determined by Hartigan's Dip-test. During training, a Dip-guided prototypical loss and a cross-domain consensus objective jointly optimize the embeddings, ensuring intra-cluster cohesion and clear inter-cluster separation.

Extensive experiments on 40 labeled datasets from the UCR archive and a large-scale unlabeled smart meter dataset demonstrate the effectiveness of the proposed framework. DipContrast consistently outperforms non-parametric baselines, achieving the highest NMI on 29 of the 40 UCR datasets. On the smart meter dataset, it attains a Silhouette score of 0.88. These results establish DipContrast as a robust and effective solution for non-parametric clustering of time-series data.

Keywords: Deep Time-Series Clustering, Contrastive Learning, Non-Parametric Clustering, Time-Series Representation Learning, Dual-Domain Architecture.

Kurzfassung

Zeitreihendaten sind in Bereichen wie dem Finanzwesen, dem Gesundheitswesen und in Energiesystemen allgegenwärtig, wo das Aufdecken verborgener Muster für nachgelagerte Analysen unerlässlich ist. Das Clustern von Zeitreihendaten bleibt aufgrund der hohen Dimensionalität, zeitlicher Abhängigkeiten und der oft unbekannten Anzahl von Clustern (k) eine Herausforderung. Während Methoden des Deep Time-Series Clustering durch die Integration von Representation Learning und Clustering vielversprechende Ergebnisse gezeigt haben, sind die meisten bestehenden Ansätze parametrisch und stützen sich primär auf Repräsentationen im Zeitbereich.

Um diese Einschränkungen zu überwinden, stellen wir **DipContrast** vor, ein End-to-End-Framework für kontrastives Lernen zum nicht-parametrischen Deep Time-Series Clustering. DipContrast nutzt eine Dual-Domain-Architektur, um robuste latente Repräsentationen sowohl aus dem Zeit- als auch aus dem Frequenzbereich zu lernen. Das Framework arbeitet ohne Vorwissen über k, indem es die Anzahl der Cluster anfänglich überschätzt und den latenten Raum schrittweise verfeinert. Es verwendet einen prototypengesteuerten, nicht-parametrischen Mechanismus, der Kandidaten-Cluster basierend auf ihrer durch Hartigans Dip-Test ermittelten Unimodalität iterativ zusammenführt. Während des Trainings optimieren eine Dip-gesteuerte prototypische Verlustfunktion (Dip-guided prototypical loss) und ein domänenübergreifendes Konsensziel (cross-domain consensus objective) gemeinsam die Embeddings und gewährleisten so den Zusammenhalt innerhalb der Cluster sowie eine klare Trennung zwischen den Clustern.

Umfangreiche Experimente auf 40 gelabelten Datensätzen aus dem UCR-Archiv und einem großen, ungelabelten Smart-Meter-Datensatz demonstrieren die Wirksamkeit des vorgeschlagenen Frameworks. DipContrast übertrifft durchweg nicht-parametrische Baselines und erzielt auf 29 der 40 UCR-Datensätze den höchsten NMI. Auf dem Smart-Meter-Datensatz erreicht es einen Silhouette-Score von 0,88. Diese Ergebnisse etablieren DipContrast als eine robuste und effektive Lösung für das nicht-parametrische Clustern von Zeitreihendaten.

Schlüsselwörter: Deep Time-Series Clustering, Kontrastives Lernen, Nicht-parametrisches Clustering, Time-Series Representation Learning, Dual-Domain-Architektur.

Contents

1. Introduction

1.1. Motivation

Time-series data consists of observations collected sequentially over time and plays a central role in a wide range of domains, including finance, healthcare, industrial monitoring, and energy systems. Unlike static data, time-series exhibits temporal dependencies, where observations are often correlated across time. This sequential nature enables the analysis of patterns such as trends, seasonality, and temporal dynamics, making time-series data essential for understanding complex systems.

As large-scale time-series data becomes more widely available, there is an increasing demand for methods capable of automatically uncovering meaningful patterns, especially in unlabeled datasets. One fundamental task in this context is *time-series clustering*, which aims to group sequences exhibiting similar temporal behavior. Effective clustering enables the discovery of latent patterns, facilitates data summarization, and supports downstream analysis in a wide range of applications.

However, clustering time-series data poses several challenges. First, time-series are often high-dimensional and exhibit complex, non-linear temporal dependencies, leading to the well-known *curse of dimensionality*, where it becomes harder to meaningfully distinguish between objects that are similar and those that are different [1]. Second, relevant structure may manifest in both the time and frequency domains, requiring representations that capture complementary aspects of the data. Third, in many real-world scenarios, the number of underlying clusters is unknown, making it difficult to apply traditional clustering methods that require this information *a priori*. These challenges are further amplified in large-scale settings, where computational complexity becomes a limiting factor, as many classical clustering algorithms exhibit quadratic or higher complexity in the number of data points [2].

Deep clustering has emerged as a promising approach to address these limitations by combining representation learning with clustering in an end-to-end framework. By leveraging neural networks, deep clustering methods learn low-dimensional embeddings that capture meaningful structure in the data, enabling more effective grouping compared to traditional techniques [3]–[6]. These approaches are particularly well-suited for high-dimensional data, as they can model non-linear relationships and produce more discriminative and robust representations [7], [8]. Furthermore, deep clustering methods often achieve improved performance in complex settings and can incorporate mechanisms for estimating the number of clusters [9]–[11].

In the context of time-series data, a wide range of methods have been proposed within the field of Deep Time-Series Clustering (DTSC). However, existing approaches typically

operate in a parametric setting and require the number of clusters to be specified in advance. In contrast, although non-parametric clustering methods exist, they are rarely tailored to time-series data. This highlights a gap in the literature, namely the lack of a well-established framework for non-parametric deep time-series clustering. Moreover, many existing methods rely primarily on representations derived from the time domain, which may not fully capture the rich characteristics of the frequency domain.

To address these challenges, this thesis proposes **DipContrast**, a novel deep clustering framework for non-parametric clustering of time-series data. The proposed method integrates contrastive representation learning with a non-parametric clustering mechanism that eliminates the need to specify the number of clusters in advance. In this context, the term non-parametric refers specifically to the ability of the model to infer the number of clusters during training, rather than requiring it as a predefined input. By jointly learning representations in both the time and frequency domains, DipContrast captures complementary temporal characteristics, leading to more expressive embeddings.

1.2. Main Contribution

The main contribution of this thesis are summarized as follows:

1. **Non-Parametric Deep Time-Series Clustering Framework:** We introduce DipContrast, an end-to-end framework that jointly learns representations and clusters without requiring the number of clusters *a priori*.

2. **Dip-Guided Cluster Refinement with DipProtoLoss:** We propose an objective function that integrates the Dip-test into the optimization process, enabling adaptive merging of unimodal clusters and non-parametric evolution of the latent space.

3. **Comprehensive Empirical Evaluation:** We evaluate DipContrast on 40 UCR datasets, achieving the highest NMI on 29 of them, and demonstrate its effectiveness on a real-world large-scale smart meter dataset.

1.3. Structure of the Thesis

The remainder of this thesis is organized as follows:

- **Chapter 2 (Related Work):** Reviews the current landscape of deep clustering methodologies, specifically focusing on recent advancements in deep time-series clustering, time-series representation learning, and non-parametric clustering methods.

- **Chapter 3 (Preliminaries):** Introduces the fundamental concepts necessary to understand the proposed framework, including time and frequency domain contexts, time-series data augmentation strategies in both domains, contrastive learning mechanisms, and the statistical principles of Hartigan's Dip-test.

- **Chapter 4 (Proposed Framework):** Provides a comprehensive breakdown of the **DipContrast** architecture, detailing the dual-domain representation learning, the Dip-guided prototypical loss, and the non-parametric cluster merging process.
- **Chapter 5 (Experiments):** Presents extensive empirical evaluations of DipContrast against non-parametric baselines, utilizing 40 labeled UCR datasets and the large-scale, unlabeled smart meter dataset. This chapter also includes qualitative analyses and ablation studies.
- **Chapter 6 (Conclusion):** Summarizes the findings, discusses its practical implications, and outlines potential avenues for future work.

2. Related Work

Rapid advancements in deep learning technologies in recent years have significantly transformed and accelerated progress in deep clustering, prompting a great deal of interest from researchers. As a result, significant efforts have been undertaken to create novel architectures and innovative deep clustering frameworks. To systematically explore this field's diverse methodologies and applications, it is essential to understand the taxonomy of deep clustering.

Various studies have proposed a range of different taxonomies. Ren et al. [12] provide a taxonomy based on the data sources and initial conditions, whereby they categorize deep clustering based on *single-view clustering*, *multi-view learning*, *semi-supervised learning*, and *transfer learning*. Single-view methods use approaches like DNNs, AEs, and GANs for feature extraction, whereas multi-view methods leverage the complementary information contained in multi-view data. Semi-supervised approaches integrate unsupervised clustering loss with constraint loss, whereby some novel techniques incorporate pairwise constraints in the feature learning process, and learn feature representations by alternatively using labeled and unlabeled data. Meanwhile, transfer learning utilizes information from relevant tasks to improve clustering performance.

On the other hand, a recent publication by Wei et al. [13] categorized deep clustering into five distinct groups based on their underlying model architectures. *DNN-based* methods utilize deep neural networks for feature extraction, integrating techniques such as iterative optimization, self-supervised learning, and contrastive learning. *AE-based* methods leverage autoencoders for dimensionality reduction and representation learning, employing both separate and joint optimization strategies. Separate optimization entails independently optimizing feature learning and clustering tasks, whilst joint optimization addresses both simultaneously to achieve a unified optimization process. *VAE-based* methods focus on using generative models to capture the underlying data distributions, with innovations like LTVAE [14] and MVAE [15] that address challenges such as multilevel clustering. *GAN-based* methods employ generative adversarial networks to learn meaningful feature representations and perform data clustering, categorized into single methods that rely solely on GANs and hybrid methods combining GANs with classifiers, AEs, or VAEs. *GNN-based* techniques focus on clustering graph-structured data, capturing both node attributes and structural relationships to deliver robust results.

Moreover, another recent study [16] presented a taxonomy based on the interaction between representation learning and clustering modules. *Multistage deep clustering* connects these modules sequentially, with representation learning providing embeddings that are later clustered using traditional methods. *Iterative deep clustering* enhances this by iteratively refining representations and clustering assignments, using clustering results (e.g., pseudo labels) to guide representation learning and vice versa. *Generative*

deep clustering leverages the transfer of gradients between the representation learning and clustering modules. The model architectures are based on VAEs and GANs with clustering approaches such as GMM to handle complex, nonlinear data distributions effectively. *Simultaneous deep clustering* jointly optimizes representation and clustering under a unified objective, employing techniques such as self-training, mutual information maximization, and contrastive learning.

Such a diverse taxonomy as well as various novel deep clustering frameworks are testament to the richness of this field of research. However, as pointed out by Zhou et al. [16], the four widely studied data types are: **image**, **text**, **video**, and **graph**, which clearly illustrates that research efforts for time-series data have not been prioritized. Given this context, we summarize the current state-of-the-art of Deep Time-Series Clustering (DTSC) and time-series representation learning.

2.1. Time-Series Representation Learning for Clustering

Learning meaningful representations is a fundamental challenge in time-series analysis due to characteristics such as temporal dependencies, high dimensionality, noise, and non-stationarity [17]. This challenge is further amplified in unsupervised settings, where the absence of labels makes it difficult to extract informative features suitable for downstream tasks such as clustering [18]. Consequently, recent advances in time-series representation learning have focused on developing unsupervised and self-supervised methods that can learn robust embeddings.

A prominent line of research is based on contrastive learning, which has emerged as a powerful paradigm for representation learning without labeled data. Methods such as TS-TCC [18], TS2Vec [19], TimesURL [20], and InfoTS [21] leverage multiple augmented views of time-series data to learn invariant and discriminative representations. These approaches typically rely on carefully designed augmentation strategies and contrastive objectives to ensure that representations of similar samples (or views) are close in the latent space, while dissimilar ones are pushed apart. Such frameworks have demonstrated strong performance across various downstream tasks, including classification and clustering, due to their ability to capture both local temporal patterns and global contextual information.

Beyond contrastive learning, other approaches focus on designing specialized encoder architectures for time-series data. For instance, convolutional encoders with dilated causal convolutions [22] have been proposed to efficiently capture long-range temporal dependencies while producing fixed-length representations for variable-length sequences. Additionally, methods such as the bilinear temporal-spectral fusion framework [23] aim to jointly model temporal and spectral characteristics of time-series, addressing limitations of approaches that rely solely on temporal information.

Despite these advancements, most existing methods are primarily designed for general-purpose representation learning and are typically evaluated on supervised or semi-supervised downstream tasks. As a result, they often do not explicitly account for the requirements of clustering, such as the formation of well-separated and compact clusters in the latent space. This limitation is closely related to what is commonly

described in deep clustering as a decoupled pipeline, where representation learning is treated as an independent preprocessing step followed by clustering with conventional algorithms such as k-means. Such an approach has been shown to be suboptimal, as the learned representations are not explicitly optimized for clustering objectives [24], [25]. This limitation is particularly pronounced in time-series data, where complex temporal dependencies, noise, and high feature correlation further complicate the learning of clustering-friendly representations [26].

Furthermore, many representation learning approaches in the contrastive learning domain rely on data augmentation strategies, the design of which can be non-trivial and domain-dependent, particularly for complex real-world data such as smart meter time-series. These limitations highlight the need for approaches that more tightly integrate representation learning with clustering, particularly for time-series data.

Historically, pioneering works in the computer vision domain successfully addressed the pitfalls of decoupled pipelines by jointly optimizing representation learning and clustering, demonstrating that aligning these objectives leads to more discriminative and cluster-friendly feature spaces [27], [28]. In the following section, we review existing DTSC methods that adopt this joint optimization paradigm to address these challenges.

2.2. Deep Time-Series Clustering

DTSC methods aim to jointly learn representations and cluster assignments in an end-to-end manner. By aligning representation learning with clustering objectives, these approaches seek to produce latent spaces that are inherently more suitable for clustering tasks. Early work in this direction includes the Deep Temporal Clustering (DTC) framework [29], which integrates a temporal autoencoder with a clustering objective to jointly learn representations and cluster assignments. Building on this idea, a large body of DTSC methods has emerged, most of which follow a joint optimization paradigm, combining reconstruction-based objectives with clustering-specific losses. As highlighted in prior surveys [26], these approaches predominantly rely on autoencoder-based architectures, including convolutional and recurrent variants, to capture temporal dependencies in the data.

A common design pattern in DTSC methods is the integration of clustering-oriented loss functions into the representation learning process. For instance, methods such as DTIC [9] iteratively refine pseudo-labels obtained from clustering algorithms to guide representation learning, thereby improving both feature quality and clustering performance. Similarly, recent works incorporate self-supervised and contrastive learning objectives into DTSC frameworks to enhance representation robustness. For example, Deep Temporal Contrastive Clustering (DTCC) [30] combines reconstruction, clustering, and contrastive losses to learn discriminative temporal representations from multiple views of the data.

Another line of research focuses on designing cluster-aware representations that explicitly encode temporal structures. For example, recent approaches incorporate additional structural constraints, such as topological information, into the learning process to better align representations with the underlying temporal patterns of the data [31].

These methods aim to move beyond purely reconstruction-driven objectives by enforcing similarity between time-series instances that share common temporal characteristics.

Despite these advancements, several limitations remain. First, many DTSC methods still rely on predefined clustering algorithms, most commonly k-means [26], which requires the number of clusters to be known *a priori* and may limit flexibility in real-world applications. Second, the heavy reliance on autoencoder-based architectures suggests that many approaches primarily focus on reconstruction quality rather than explicitly optimizing for clustering performance.

Overall, DTSC methods represent a significant step toward bridging the gap between representation learning and clustering for time-series data. However, most existing approaches remain parametric, requiring prior knowledge of the number of clusters and often relying on heuristics to determine this parameter. This limitation naturally motivates the exploration of non-parametric deep clustering methods, which aim to automatically infer the number of clusters while maintaining strong representation learning capabilities.

2.3. Evolution of Non-Parametric Clustering

As established, the reliance on a predefined number of clusters, k, is a fundamental drawback of most existing deep clustering approaches. In many real-world applications, this assumption is impractical because the true underlying distribution is unknown and difficult to estimate reliably.

Early foundational work to address this issue focused on extending traditional algorithms like k-means to dynamically determine the optimal k. For instance, Pelleg and Moore introduced X-means [32], which extends k-means by iteratively splitting clusters and selecting the optimal number using the Bayesian Information Criterion (BIC). To improve efficiency, the method leverages a multiresolution kd-tree structure, enabling fast local decisions on whether clusters should be divided.

Hamerly and Elkan proposed G-means [33], which determines the number of clusters by testing whether the data within each cluster follows a Gaussian distribution. This is achieved using the Anderson-Darling test on projected data, allowing the algorithm to iteratively refine cluster splits. Building on this idea, PG-means [34] evaluates the fit of a Gaussian mixture model to the entire dataset using the Kolmogorov-Smirnov test. By assessing the model globally rather than per cluster, it is more robust in scenarios with overlapping or irregular cluster structures.

To relax the strong Gaussian assumption, Kalogeratos and Likas introduced Dip-means [35], which only assumes that each cluster is unimodal. The method applies Hartigan's dip-test to determine whether a cluster should be split, based on the distribution of pairwise distances within the cluster. This allows Dip-means to better handle non-Gaussian and irregular cluster shapes compared to earlier approaches.

More recent approaches integrate non-parametric principles with deep learning. Ye et al. [36] propose a Bayesian framework based on Dirichlet Process Mixture Models, where deep neural networks model complex data manifolds and a VAE enables efficient inference. This approach allows clustering without predefined k and can adapt to diverse

data distributions, although it introduces additional model complexity.

Subsequent work focuses on combining deep representation learning with non-parametric clustering objectives. Ren et al. [4] introduce DDC, a two-stage framework that first learns representations via a DCAE and then applies density-based clustering to infer the number of clusters. DeepDPM [37] further advances this idea by integrating a split-and-merge mechanism into a neural network, dynamically adjusting k during training and improving scalability.

Another line of research explores hierarchical and merge-based strategies. DeepECT [38] constructs a binary cluster tree in the latent space without requiring prior knowledge of k, capturing hierarchical relationships in the data. Similarly, DipDECK [11] combines autoencoder-based embeddings with iterative merging guided by Hartigan's dip-test, enabling flexible clustering of non-convex structures. These methods, however, are primarily developed for data such as images and do not incorporate time-series-specific representation learning techniques.

2.4. Research Gap: Non-Parametric Deep Time-Series Clustering

While the broader field of non-parametric deep clustering is well-established, the majority of the methods reviewed rely exclusively on image data for their evaluation. In fact, to the best of our knowledge, there is currently no work that explicitly combines non-parametric deep clustering with time-series-specific representation learning in a unified framework. Simultaneously, while representation learning and parametric DTSC have matured significantly for time-series data, the absence of non-parametric DTSC remains a critical research gap. The proposed framework, DipContrast, is specifically designed to sit directly at this intersection, integrating dual-domain time-series representation learning with non-parametric cluster evolution to address this gap.

3. Preliminaries

This chapter describes the fundamental concepts and architectural components that constitute the proposed framework. We first give a short introduction to time-series data and its domains, as well as the contrastive learning framework. Then, we discuss time-series augmentation strategies in the context of contrastive learning, followed by the neural networks and relevant architectures employed for extracting meaningful latent representation of time-series data. Finally, we provide brief description of Hartigan's dip test, which evaluates the unimodality of candidate clusters and governs the logic of iterative cluster merging process in the proposed framework.

3.1. Time-Series Fundamentals and Domain Contexts

A time-series is fundamentally a sequence of data points recorded at successive, and often uniform, intervals of time. In formal terms, a time-series can be defined as an ordered set of observations $X = \{x_1, x_2, \ldots, x_T\}$, where each observation x_t is recorded at a specific timestamp t. Depending on the complexity of the underlying system being monitored, x_t can be a scalar value $x_t \in \mathbb{R}$ (univariate time-series) or a vector $x_t \in \mathbb{R}^d$ (multivariate time-series), where d represents the number of distinct variables measured simultaneously.

Time-series data is characterized by its inherent temporal dependence. The value of an observation at time t is highly correlated with its historical values at $t-1, t-2, \ldots, t-k$. This temporal continuity means that the sequential ordering of the data carries critical semantic information. Analyzing this data requires frameworks capable of handling high dimensionality, non-stationarity, and varying levels of background noise, while successfully extracting the underlying patterns that govern the data generation process. To effectively capture these underlying patterns, time-series data is commonly analyzed through different representational lenses, or "domains." Transforming the data into different domains does not alter the underlying information, rather it reorganizes the data to make specific structural characteristics more accessible to analytical models and machine learning algorithms. The two most prominent domains are the time domain and the frequency domain.

The Time Domain (x^t) The time domain represents a time-series in its original sequential form, where observations are expressed directly as a function of time, x_t versus t. In this domain, analysis focuses on how the information evolves over successive timestamps, preserving the temporal ordering and dependencies inherent in the data. Patterns such as trends, seasonality, abrupt changes, autocorrelation, and temporal dynamics are examined directly through the progression of values in time. Methods operating in the time domain

model the relationships between present and past observations, capturing short-term and long-term dependencies while accounting for noise and non-stationary behavior.

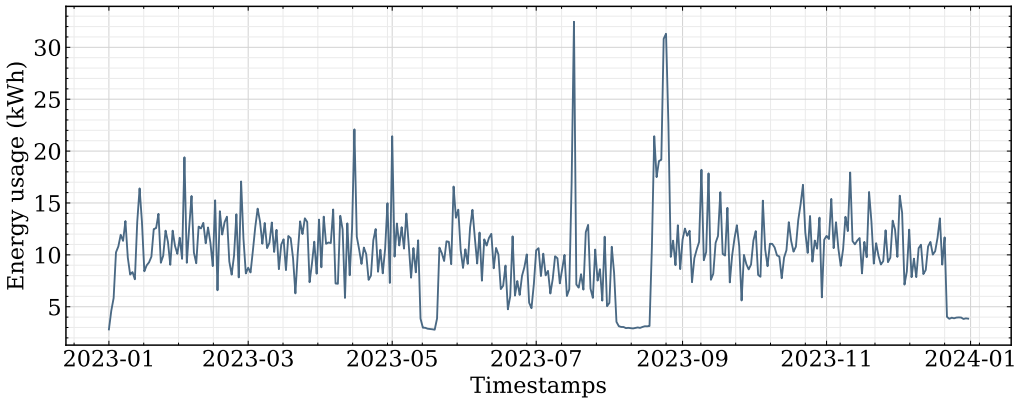


Figure 3.1.: Time domain profile of daily energy consumption (kWh).

The Frequency Domain (x^f) The frequency domain represents a time-series by decomposing it into a set of sinusoidal components characterized by distinct frequencies, amplitudes, and phases. Instead of describing how the data changes over time, this domain describes how much of its energy or variance is distributed across different frequency components. Through mathematical transformations such as the Fourier transform, the original time-ordered sequence is re-expressed in terms of periodic structures, making recurring cycles, oscillatory behavior, and hidden periodicities more explicit. The frequency domain is especially valuable for identifying dominant rhythms, filtering noise, and analyzing data where periodic patterns are more informative than their exact timing in the time sequence.

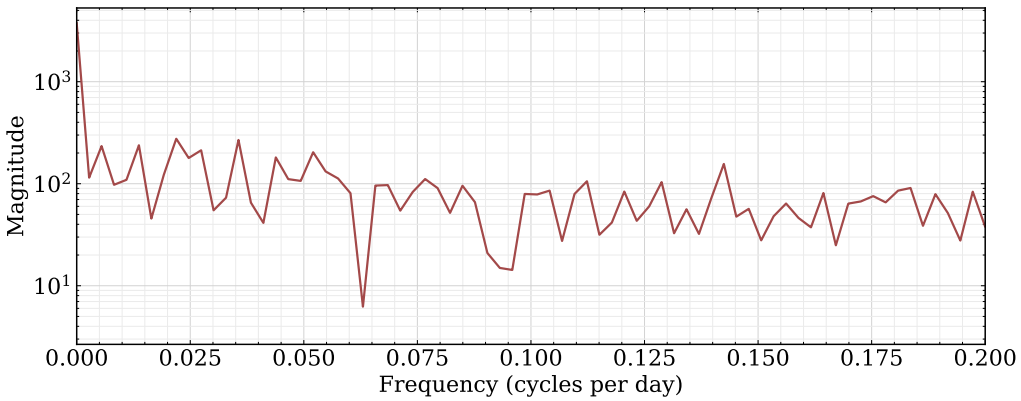


Figure 3.2.: Frequency spectrum of energy usage.

In the context of smart meter data, relying solely on a single domain provides an

incomplete picture of consumer behavior. Smart meters record energy consumption at frequencies such as every 15 or 30 minutes, resulting in multifaceted, non-stationary data streams. While analyzing this data in the time domain is essential for capturing localized, short-term events (such as intra-day consumption peaks, transient load fluctuations, and sudden anomalous energy draws), relying only on this temporal view limits the analysis. As smart meter data is deeply embedded in broader contextual cycles, leveraging the frequency domain allows for the discovery of latent periodicities, such as multi-day operational cycles and broader seasonal variations in energy demand.

Therefore, a dual-domain approach is necessary, and by extracting features from both domains, machine learning models can construct a much richer, more holistic latent representation of time-series data. This comprehensive approach ensures that neither temporal anomalies nor fundamental cyclical patterns are overlooked during the clustering and analysis phases.

3.2. Contrastive Learning

Contrastive learning sits within the broader realm of Self-Supervised Learning (SSL), where the objective is to learn representations from raw, unlabeled data by identifying inherent patterns and relationships within the data itself. The primary motivation is pragmatic and statistical, as manual labeling is expensive, requires domain expertise, and often infeasible at scale, while unlabeled data are abundant. The prominence of contrastive learning stems from its core objective, that is aligning the representations of similar data points while pushing dissimilar ones apart.

Training Mechanism of the Contrastive Learning Framework At a high level, the objective of the training mechanism is to minimize the distance between *positive pairs*, while maximizing the distance between *negative pairs*. Formally, given an unlabeled dataset, an encoder network f_θ is trained to map inputs into a latent representation space. For any anchor data point x_i, a contrastive algorithm constructs a positive sample x_i^+ and a set of negative samples $\{x_1^-, x_2^-, \ldots, x_K^-\}$. The encoder projects these inputs to yield the respective embeddings z_i, z_i^+, and $\{z_1^-, z_2^-, \ldots, z_K^-\}$. To guide the training mechanism of the contrastive learning framework, a broad spectrum of loss functions has been proposed in the literature including contrastive loss [39], triplet loss [40], multi-class N-pair loss [41], NCE loss [42], and InfoNCE loss [43]. Given its utilization within the proposed framework, we provide a brief outline of the InfoNCE loss below.

The InfoNCE loss is effectively a categorical cross-entropy problem formulated to identify the single positive sample among a pool of negatives. Mathematically, the loss is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\exp(\text{sim}(z_i, z_i^+)/\tau) + \sum_{k=1}^{K} \exp(\text{sim}(z_i, z_k^-)/\tau)}$$

where $\text{sim}(\cdot, \cdot)$ denotes a similarity metric, conventionally the cosine similarity $\text{sim}(u, v) = u^\top v/(\|u\|\|v\|)$, and τ is a temperature scaling hyperparameter that dictates the penalty

applied to hard negative samples. By minimizing this loss, the model maximizes a lower bound on the mutual information between the positive pairs. As the network undergoes continuous gradient updates, the geometry of the embedding space is shaped through a fundamental *push-and-pull* mechanism. The encoder is driven to pull together and tightly cluster the vectors of related instances while simultaneously pushing away and dispersing the representations of unrelated, negative instances across the latent space. This push-and-pull dynamic ensures the resulting latent space accurately reflects underlying semantic similarity.

The Role of Data Augmentation In an unsupervised environment, defining a *"positive pair"* is a non-trivial challenge due to the absence of labels. Contrastive learning resolves this through stochastic data augmentation as the main mechanism for generating positive samples. By applying a predefined family of transformations $\mathcal{T}$ to a single instance x, multiple distinct, augmented views are generated and treated as positive samples of x. The network is incentivized to learn invariant semantic representations in order to maximize agreement between these augmented views in the latent space. Consequently, the quality of the learned representations depends heavily on the specific augmentation strategy employed. Weak augmentations allow the network to minimize the loss by relying on trivial features, while overly aggressive transformations risk discarding the core semantic identity. Furthermore, the choice of appropriate data augmentation techniques is dictated by the modality of the data (e.g., image, time-series) and the downstream task. Given that data augmentation plays a pivotal role in the contrastive learning, we outline the time-series augmentation methods utilized in the proposed framework in section 3.3.

Contrastive Learning Architectures As contrastive learning has rapidly matured, several distinct architectural philosophies have been proposed including SimCLR [44], MoCo [45], BYOL [46], SwAV [47], and SimSiam [48]. As the proposed framework directly utilizes the end-to-end joint optimization strategy pioneered by SimCLR, we briefly outline its underlying architecture here.

Among the wide array of contrastive frameworks, Simple Contrastive Learning of Representations (SimCLR) is widely recognized for its minimalist yet profoundly effective design. The SimCLR architecture consists of four sequential components. First, a stochastic data augmentation module generates augmented views of the same batch of images. SimCLR experimentally demonstrated that the composition of random cropping and aggressive color distortion is uniquely critical to its success, as it prevents the model from relying on local color patches. Second, a base neural network encoder $f(\cdot)$, universally a standard ResNet architecture, extracts representation vectors $h_i = f(x_i)$ from the augmented views. Third, SimCLR introduces a crucial architectural innovation: a non-linear projection head $g(\cdot)$, typically a Multilayer Perceptron (MLP) with one hidden layer and a ReLU activation, maps the representations into a lower-dimensional latent space $z_i = g(h_i)$. The contrastive loss is applied exclusively to z_i rather than the intermediate representation h_i. This projection head effectively isolates the contrastive objective, allowing h_i to retain generalized, rich information. Finally, the framework

employs the Normalized Temperature-Scaled Cross-Entropy (NT-Xent) loss (a specific variant of the widely adopted InfoNCE objective) to maximize agreement between positive pairs. SimCLR operates on the current mini-batch, and for a batch size of N, the algorithm generates $2N$ augmented views, utilizing the remaining $2(N-1)$ views as negative samples for any given pair.

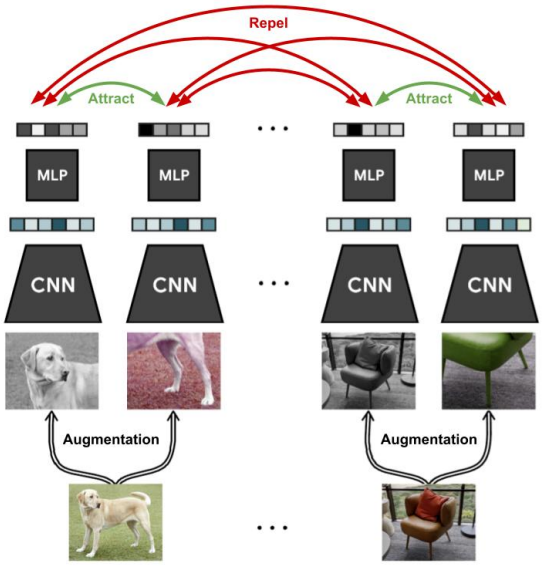


Figure 3.3.: SimCLR learning framework. [44]

While SimCLR provides a highly effective, end-to-end architecture, it introduces significant trade-offs. Its primary advantage lies in its conceptual simplicity and streamlined training pipeline, which integrates an augmentation pipeline with an end-to-end contrastive optimization objective to effectively acquire robust and discriminative latent representations. However, SimCLR's reliance on in-batch negative sampling creates a computational bottleneck. To prevent representational collapse and provide a sufficient number of negative samples, the framework requires large batch sizes (often $N \geq 4096$ as suggested by [44]). Although careful hyperparameter optimization can enable smaller batch sizes to achieve performance comparable to those utilizing large batch sizes [49].

3.3. Time-Series Data Augmentation

The data augmentation methods utilized in the proposed framework are described as follows, categorized by their respective domain of application:

3.3.1. Time Domain Data Augmentation

Given a time-series x_i^t in the temporal domain, the augmented data $\tilde{x}_i^t$ is generated by applying an operation from a predefined set of augmentation methods. A number

of augmentation methods have been proposed including *jittering*, *scaling*, *TimeWarp*, *MagWarp*, *cropping* as well as combination of several of these methods [50]. Within the proposed DipContrast framework, we utilize three data augmentation techniques for the time domain. These augmentations are formulated as $\tilde{x}_i^t = T^j(x_i^t)$, where $j \in \{a, b, c\}$, and are described as follows:

Jittering $T^a(x_i^t)$**:** Jittering is the process of perturbing a time-series by adding random Gaussian noise to each individual observation, effectively simulating sensor noise. The augmented series $\tilde{x}_i^t$ is defined as:

$$\tilde{x}_i^t = x_i^t + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{jitter}})$ is a noise tensor of the same shape as x_i^t, drawn from a normal distribution with a mean of 0 and a standard deviation defined by the hyperparameter σ_{jitter}.

Scaling $T^b(x_i^t)$**:** Scaling alters the global magnitude of the time-series without changing its fundamental shape or temporal dynamics. The scaling factor is applied per feature but remains constant across the entire sequence length for a given sample. The mathematical formulation is:

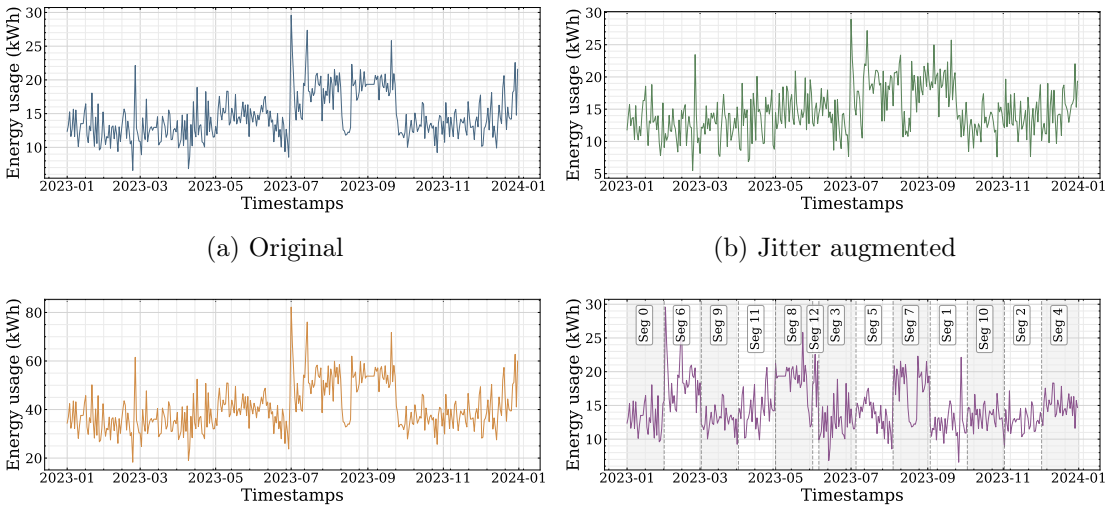
$$\tilde{x}_i^t = \alpha \cdot x_i^t$$

where $\alpha \sim \mathcal{N}(1, \sigma^2_{\text{scaling}})$ is the scaling factor drawn from a Gaussian distribution centered at 1.0 with a standard deviation of σ_{scaling}.

Permutation $T^c(x_i^t)$**:** Permutation breaks the time-series into discrete local windows and randomly rearranges the order of these windows. Given a sequence of length L, it is divided into $K = \lceil L/W \rceil$ segments, where W is the predefined window size. Importantly, if the sequence length L is not perfectly divisible by the window size W, the final segment s_K will simply contain the remaining elements and be smaller than the defined window size. If the ordered set of segments is $S = \{s_1, s_2, \ldots, s_K\}$, the augmented time-series is formed by shuffling them:

$$\tilde{x}_i^t = \{s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(K)}\}$$

where π represents a random permutation of the segment indices $\{1, 2, \ldots, K\}$.



(a) Original

(b) Jitter augmented

(c) Scaling augmented

(d) Permutation augmented

Figure 3.3.: Original and augmented views of a smart meter profile in time-domain.

3.3.2. Frequency Domain Data Augmentation

For a given frequency spectrum x_i^f, the augmented representation $\tilde{x}_i^f$ is produced by executing a transformation selected from a designated set of augmentation strategies. Following the approach established in [51], our proposed framework incorporates two data augmentation methods specifically designed for the frequency domain. These augmentations are formulated as $\tilde{x}_i^f = F^j(x_i^f)$, where $j \in \{a, b\}$, and are described as follows:

Add Frequency $F^a(x_i^f)$: This augmentation methods targets frequency components with amplitudes below a certain threshold and artificially boosts them. First, we determine the maximum amplitude in the frequency spectrum, $A_m = \max(x_i^f)$, and define the target threshold as ωA_m, where ω is the predefined scaling factor. Next, we define a binary mask M of the same shape as x_i^f. An element in M is set to 1 if the corresponding frequency amplitude is smaller than the threshold and it is selected by the random perturbation rate θ. Let $U \sim \mathcal{U}(0, 1)$ be a tensor of random variables drawn from a uniform distribution between 0 and 1. The mask is defined as:

$$M = \mathbb{I}(x_i^f < \omega A_m \wedge U < \theta)$$

where $\mathbb{I}(\cdot)$ is the indicator function returning 1 if the condition is true and 0 otherwise. The augmented frequency spectrum $\tilde{x}_i^f$ is then calculated by applying this mask to replace the selected components with the threshold value:

$$\tilde{x}_i^f = x_i^f \odot (1 - M) + (\omega A_m) \cdot M$$

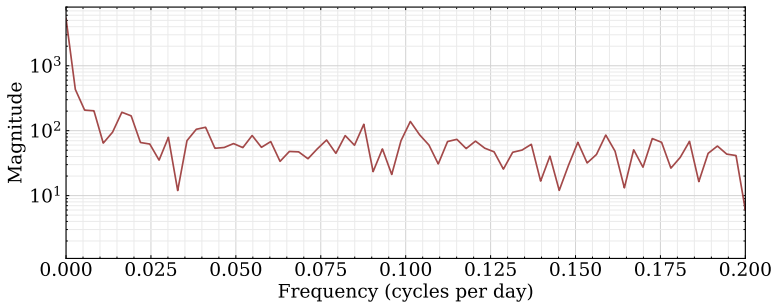
where $\odot$ represents the Hadamard (element-wise) product.

Remove Frequency $F^b(x_i^f)$: Remove frequency randomly discards (zeros out) existing frequency components based on a masking rate ϵ. Let $V \sim \mathcal{U}(0,1)$ be a tensor of random variables drawn from a uniform distribution between 0 and 1, with the same shape as x_i^f. We define a binary mask M' where elements are kept (1) only if the random variable exceeds the masking rate ϵ:

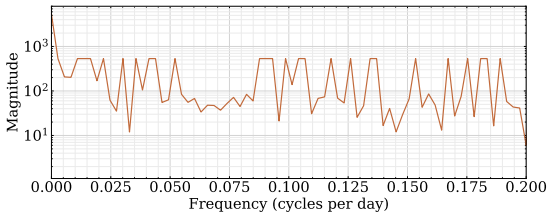
$$M' = \mathbb{I}(V > \epsilon)$$

The augmented frequency spectrum $\tilde{x}_i^f$ is obtained by applying this mask directly to the input spectrum:

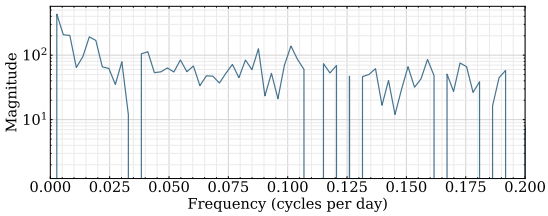
$$\tilde{x}_i^f = x_i^f \odot M'$$



(a) Original



(b) Add frequency augmented



(c) Remove frequency augmented

Figure 3.4.: Original and augmented views of a smart meter frequency-domain profile.

3.4. Neural Networks

Deep Neural Networks (DNNs) are highly effective at mapping sequential data into non-linear latent manifolds. These architectures utilize stacked hidden layers to encode raw, high-dimensional time-series data into compact, dense vector embeddings. This non-linear transformation captures both short- and long-range temporal dependencies, non-stationary trends, and multi-scale periodicities, yielding robust feature vectors optimized for down-stream tasks such as clustering. Among the various deep neural network architectures available, we briefly describe bidirectional Long Short-Term Memory (Bi-LSTM) and

one-dimensional Convolutional Neural Network (1D CNN), which are employed within the proposed framework to learn latent representations of time-series data in different domain.

3.4.1. Bidirectional LSTM

Recurrent Neural Networks (RNNs) were among the first architectures utilized for modeling sequential data such as time-series. RNNs maintain a hidden state that is updated at each timestep, theoretically allowing them to retain a memory of past inputs. However, vanilla RNNs face challenges when training on long sequences due to the vanishing gradient problem, which prevents the network from learning long-term dependencies. Long Short-Term Memory (LSTM) networks were introduced to address this limitation. The key architectural addition is a persistent cell state c_t regulated by dedicated gating mechanisms. The cell state acts as an internal conveyor belt, allowing information to flow down the sequence with minimal linear interactions, thereby preserving gradients over extended temporal distances.

The flow of information into and out of the cell state is regulated by three distinct gates: the forget gate, the input gate, and the output gate. Given an input sequence observation x_t at timestamp t and the hidden state from the previous timestamp h_{t-1}, the standard LSTM operates through a precise sequence of mathematical operations. First, the forget gate controls what information is discarded from the internal state:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Here, W_f and U_f are learnable weight matrices, b_f is the bias vector, and σ is the sigmoid activation function. Next, the cell must decide what new information to store. The input gate identifies which specific values in the memory state will be updated:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

Parameterized by the weight matrices W_i, U_i and bias b_i, this input gate works in tandem with a parallel mechanism that generates new candidate values:

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

The candidate state $\tilde{c}_t$ is created using the hyperbolic tangent function tanh, squashing the new values between -1 and 1 utilizing its own set of weights W_c, U_c and bias b_c. With both the forget and input mechanisms established, the old cell state c_{t-1} is updated to the new cell state c_t:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

In this state update, the previous state c_{t-1} is scaled by the forget gate f_t to effectively drop targeted information, and the result is added to the new candidate state $\tilde{c}_t$ scaled by the input gate i_t.

Finally, the cell determines its output for the current timestep. An output gate dictates which parts of the newly updated cell state will be exposed as the final output:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

The gate o_t relies on the weight matrices W_o, U_o and bias b_o. The new hidden state is then calculated based on that output gate:

$$h_t = o_t \odot \tanh(c_t)$$

The hidden state h_t is obtained by pushing the updated cell state c_t through a tanh function and filtering it via elementwise multiplication with the output gate o_t, before passing it to the next timestep.

Building on this single-direction foundation, the Bidirectional LSTM processes the data sequence in both chronological and reverse orders to capture a richer temporal context. It maintains two independent LSTM layers, yielding a sequence of forward hidden states $\{\overrightarrow{h}_t\}$ and backward hidden states $\{\overleftarrow{h}_t\}$. At any given timestep t, the final Bi-LSTM representation is formed by combining these two states:

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t]$$

The concatenation $[;]$ of the forward vector $\overrightarrow{h}_t$ and backward vector $\overleftarrow{h}_t$ merges the dual representations. Assuming each directional LSTM has a hidden dimensionality of d, the resulting vector h_t belongs to the space $\mathbb{R}^{2d}$, yielding a unified feature representation that captures both past and future temporal contexts simultaneously.

3.4.2. 1D CNN

While Convolutional Neural Networks (CNNs) originated in the field of computer vision to efficiently extract hierarchical features from images, the underlying principle can be applied just as effectively to one-dimensional sequential data. A 1D CNN shifts learnable filters along the temporal axis to detect key motifs independently of their position in time.

In formal terms, when processing a multivariate time series through stacked network layers, a 1D convolutional layer applies a set of learnable filters across the input channels. Let M_{l-1} denote the number of input channels received from the previous layer $l-1$. The j-th output feature map at layer l and discrete time step n is computed as:

$$y_j^{(l)}[n] = f\left(\sum_{i=1}^{M_{l-1}} (x_i^{(l-1)} * w_{ij}^{(l)})[n] + b_j^{(l)}\right)$$

where $*$ denotes the 1D discrete convolution operation along the temporal axis, and $x_i^{(l-1)}$ represents the i-th input channel sequence from the preceding layer. The term $w_{ij}^{(l)}$ is the learnable convolutional kernel (filter weights) connecting input channel i to the output feature map j, while $b_j^{(l)}$ is the learnable bias term specific to the j-th filter.

Finally, $f(\cdot)$ applies a nonlinear activation function (e.g., ReLU). By aggregating these convolutions across all M_{l-1} input channels, the layer produces a new set of multi-channel outputs. Each learned filter functions as a detector for a particular temporal pattern across channels, enabling the network to build increasingly abstract representations as layers are stacked.

3.5. Hartigan's Dip Test

Hartigan's dip test, introduced by Hartigan and Hartigan [52], provides a statistical measure of unimodality. In statistics, unimodality refers to a data distribution that has a single, distinct peak or mode. The test takes one-dimensional data as input, and computes the corresponding dip value (or dip statistic). The dip value is a nonnegative real number that lies in the interval $[0, 0.25]$. A value of 0 indicates that the empirical distribution is perfectly consistent with unimodality, meaning that no deviation from a unimodal cumulative distribution function is required. As the dip value increases toward its theoretical upper bound of 0.25, the empirical distribution exhibits increasingly pronounced departures from unimodality, corresponding to clearer separation between multiple modes. Accompanying the dip value is the dip p-value, which takes values in the interval $[0, 1]$. The p-value quantifies the statistical evidence against the null hypothesis of unimodality. A p-value close to 1 indicates that the observed dip value is highly consistent with a unimodal distribution, whereas a p-value close to 0 suggests strong evidence of multimodality. In standard hypothesis testing practice, a small p-value (e.g., below a pre-specified significance level such as 0.05) leads to rejection of the null hypothesis, implying that the marginal distribution of the time-series observations is unlikely to be unimodal.

Within non-parametric clustering frameworks, the dip statistic can serve as a decision mechanism for evaluating whether candidate clusters should be merged. Specifically, it assesses whether the combined distribution of two clusters is consistent with unimodality. A relatively large p-value indicates that merging the clusters is statistically justified. In contrast, a small p-value provides evidence against unimodality.

4. Proposed Framework

The fundamental challenge in unsupervised time-series clustering is discovering meaningful semantic partitions in data when ground truth or the true number of clusters (k) is unknown *a priori*. To address this challenge, we propose **DipContrast**, a contrastive learning framework for non-parametric deep time-series clustering. It draws inspiration from three core concepts: dual-domain time-series representation learning from CDCC [51], prototype based contrastive clustering from PCL [53], and non-parametric clustering from DipDECK [54]. At its core, DipContrast is an end-to-end deep clustering framework that unifies representation learning and clustering by shaping latent embeddings to be invariant across augmented views and consistent across domains, while simultaneously organizing the latent space via prototypes that adaptively merge when unimodality tests indicate they represent a single cluster.

4.1. Dual-Domain Contrastive Representation

The architecture for latent representation learning in DipContrast employs a dual-domain network inspired by CDCC. To construct the latent space, first the framework generates two domain-specific views from the input time-series instance. It subsequently applies targeted augmentations to yield contrastive pairs. Both the original time-series instances and their corresponding augmented instances are then passed through dedicated, domain-specific encoders. DipContrast facilitates fully unsupervised representation learning through dynamic, learnable prototypes, which are initialized by subjecting the framework to an instance-level contrastive warm-up prior to clustering.

4.1.1. Feature Extraction and Instance Projection

The framework relies on two separate neural network encoders to extract features or representations, with each encoder explicitly designed to accommodate the specific structural characteristics of its corresponding input modality.

Time Encoder The time encoder is implemented as a Bidirectional Long Short-Term Memory (LSTM) network designed to process time-domain sequences. The encoder effectively extracts abstract features across various time periods by processing sequences in both directions, capturing both past and future information. The outputs from Bi-LSTM are subsequently passed through a dropout layer to mitigate overfitting, and mapped via a linear projection layer to form the initial representation.

Frequency Encoder The frequency encoder processes frequency-domain input sequences using a 1D Convolutional Neural Network (CNN) architecture. This encoder is composed of three sequential convolutional blocks. The initial block applies a 1D convolution with a configurable kernel size, stride, and padding, followed by Batch Normalization, a ReLU activation function, a 1D Max Pooling operation to downsample the temporal resolution, and a dropout layer. The subsequent two blocks repeat the sequence of convolution, normalization, activation, and pooling to extract latent representation for frequency domain.

Let X^d represent a batch of input data for a specific domain $d \in \{t, f\}$, where t denotes the temporal view and f denotes the frequency view obtained via *Fast Fourier Transform*. The framework applies a series of augmentations to generate augmented views, denoted as $\tilde{X}^d$. Crucially, this augmentation process operates independently on each individual instance within the batch rather than uniformly across the entire batch. The framework applies a single augmentation method to each input, sampled stochastically from a domain-specific pool according to a predefined probability distribution. For the temporal domain, an individual input $x_i^t \in X^t$ undergoes either *jittering* $T^a(x_i^t)$, *scaling* $T^b(x_i^t)$, or *permutation* $T^c(x_i^t)$, depending on the randomly sampled outcome. Likewise, for the frequency domain, a frequency input $x_i^f \in X^f$ undergoes either *add frequency* $F^a(x_i^f)$ or *remove frequency* $F^b(x_i^f)$, dictated by a separate set of selection probabilities. With the contrastive pairs generated, the multi-stage encoding and projection pipeline proceeds by processing both the original inputs X^d and the augmented inputs $\tilde{X}^d$ through their dedicated domain encoders. This initial step produces the intermediate hidden representations, H^d and $\tilde{H}^d$, where the time-domain latent representations are subjected to an additional dropout operation.

Following this intermediate extraction, both H^d and $\tilde{H}^d$ are mapped through dedicated, domain-specific instance-level projection heads. The projection heads are Multilayer Perceptrons consisting of an initial linear layer followed by a ReLU activation and a final linear layer that yields the final contrastive embedding sets $Z^d = \{z_1^d, \ldots, z_n^d\}$ and $\tilde{Z}^d = \{\tilde{z}_1^d, \ldots, \tilde{z}_n^d\}$.

4.1.2. Shift from Classification Layers to Prototypes

In CDCC, clustering is performed mainly on the time-domain, and clustering relies on a dedicated projection head acting as a static classification layer. Intermediate hidden states are processed through a two-layer multi-layer perceptron and a softmax function to generate a normalized probability distribution over a predetermined number of clusters. Finally, the discrete cluster assignment for each data instance is obtained by simply selecting the category with the highest predicted probability.

Although effective when the true number of clusters is known, requiring this parameter *a priori* introduces an issue for unsupervised setting. As a result, DipContrast adopts a purely prototype-driven architecture, and uses a set of prototypes, $C^d = \{c_1^d, \ldots, c_{k_d}^d\}$ to facilitate non-parametric clustering within the latent space.

4.1.3. Warm-up using Instance-Level Contrastive Loss

Before clustering begins, the model undergoes a *"warm-up"* phase guided by the instance-level contrastive loss. The instance-level contrastive loss ($\mathcal{L}^{\text{Ins}}$) is designed to push paired positive samples together while repelling all other negative samples within the batch. This objective ensures that the embeddings are invariant to augmentations within each domain while simultaneously aligning the latent spaces.

The instance loss relies on a temperature-scaled cross-entropy calculation operating on a similarity matrix derived from Z^d and $\tilde{Z}^d$. For a given domain instance embedding z_i^d and its augmented counterpart $\tilde{z}_i^d$, the pairwise loss is formulated to maximize their cosine similarity while minimizing the similarity against all other $2n-2$ negatives, among the $2n-1$ candidates excluding the anchor itself. The loss for a single representation is computed as:

$$\mathcal{L}_{z_i^d}^{\text{Ins}} = -\log \frac{\exp(sim(z_i^d, \tilde{z}_i^d)/\tau_{ins})}{\sum_{j=1,j\neq i}^{n} \exp(sim(z_i^d, z_j^d)/\tau_{ins}) + \sum_{j=1}^{n} \exp(sim(z_i^d, \tilde{z}_j^d)/\tau_{ins})}, \tag{4.1}$$

where τ_{ins} is the instance-level temperature parameter. The total instance loss for a given domain is the average over all n samples and their augmented views:

$$\mathcal{L}^{\text{Ins}}(Z^d, \tilde{Z}^d) = \frac{1}{2n} \sum_{i=1}^{n} \left(\mathcal{L}_{z_i^d}^{\text{Ins}} + \mathcal{L}_{\tilde{z}_i^d}^{\text{Ins}} \right). \tag{4.2}$$

During the warm-up iterations, the optimizer updates the network parameters by minimizing a composite loss function. This function balances the intra-domain instance losses against the cross-domain instance loss using a weighting hyperparameter λ:

$$\mathcal{L}^{\text{warm-up}} = \lambda(\mathcal{L}_t^{\text{Ins}} + \mathcal{L}_f^{\text{Ins}}) + (1-\lambda)\mathcal{L}_{tf}^{\text{Ins}}, \tag{4.3}$$

where the individual components are defined as:

$$\mathcal{L}_t^{\text{Ins}} = \mathcal{L}^{\text{Ins}}(Z^t, \tilde{Z}^t);\ \mathcal{L}_f^{\text{Ins}} = \mathcal{L}^{\text{Ins}}(Z^f, \tilde{Z}^f);\ \mathcal{L}_{tf}^{\text{Ins}} = \mathcal{L}^{\text{Ins}}(\tilde{Z}^t, \tilde{Z}^f).$$

This crucial step establishes a tightly aligned feature space before cluster prototypes are introduced into the optimization pipeline. It is however noteworthy, that the number of warm-up epochs should be limited. As presented in the ablation study in CDCC, instance-level contrastive loss alone performs poorly. Our goal with this warm-up step is to ensure a friendly latent space to facilitate stable prototype initialization.

4.2. Contrastive Learning with Prototypes

After the warm-up phase, DipContrast transitions to a prototype-based learning stage that functions as an alternating optimization procedure. Inspired by the Expectation-Maximization (EM) formulation of PCL, latent prototype assignments are first derived from the current embedding space and subsequently utilized to direct the next encoder

update. This procedure is executed independently across the temporal and frequency domains, yielding two distinct prototype sets that jointly guide representation learning.

Let $d \in \{t, f\}$ denote the temporal and frequency domains. For each input sample, the domain-specific encoder produces latent representations, mapping the original samples (x_i^t, x_i^f) and their augmented versions $(\tilde{x}_i^t, \tilde{x}_i^f)$ to their respective embeddings z_i^d and $\tilde{z}_i^d$. We further introduce a latent prototype variable c^d for each domain that takes values from a discrete, dynamically sized set $C^d = \{c_1^d, \ldots, c_{k_d}^d\}$.
Introducing these prototypes as latent cluster centers allows us to formulate the prototypical representation learning as maximizing the log-likelihood of the observed data:

$$\max_{\theta} \sum_{i=1}^{n} \log p(x_i^d; \theta) = \max_{\theta} \sum_{i=1}^{n} \log \sum_{c \in C^d} p(x_i^d, c; \theta). \tag{4.4}$$

As in PCL, a lower bound is obtained by introducing an auxiliary distribution $Q_i^d(c)$:

$$\sum_{i=1}^{n} \log \sum_{c \in C^d} p(x_i^d, c; \theta) \geq \sum_{i=1}^{n} \sum_{c \in C^d} Q_i^d(c) \log \frac{p(x_i^d, c; \theta)}{Q_i^d(c)}. \tag{4.5}$$

In PCL, $Q_i^d(c)$ is realized via hard cluster assignments, reducing the optimization to a rigid classification task. While DipContrast utilizes hard assignments (k-means) to establish the initial pseudo-labels during the *E-step*, it fundamentally diverges during the *M-step* by transforming these hard assignments into soft targets via the Dip-p-value matrix.

Specifically, the optimization alternates between two stages. The *E-step* performs k-means clustering to estimate the latent prototypes, calculates the prototype-wise concentration values, and conducts the Dip-test to evaluate structural affinities and merge valid micro-clusters. The *M-step* then updates the encoder parameters using the DipProtoLoss. This ensures that latent embeddings do not merely converge toward a single hard-assigned prototype, but evolve to respect the broader non-parametric cluster geometry discovered during the *E-step*.

4.2.1. E-step: Prototype and Concentration Estimation

Prior to each optimization cycle, latent representations for all samples are computed in each domain. DipContrast then runs k-means clustering independently on the full representation sets Z^t and Z^f. For each domain $d \in \{t, f\}$, this yields hard assignments $y_i^d \in \{1, \ldots, K_{\text{remaining}}\}$ and prototypes $c_1^d, \ldots, c_{K_{\text{remaining}}}^d$:

$$y_i^d = \arg \min_{1 \leq j \leq K_{\text{remaining}}} \|z_i^d - c_j^d\|_2^2, \quad c_j^d = \frac{1}{|C_j^d|} \sum_{i:\, y_i^d = j} z_i^d, \quad \text{where } C_j^d = \{i : y_i^d = j\} \tag{4.6}$$

Here, $K_{\text{remaining}}$ denotes the number of clusters remaining in the current cycle. Initially, $K_{\text{remaining}} = K_{\text{init}}$, but as training progresses, it represents the number of clusters left

after previous cluster merging operations (see subsection 4.3.4). Because both the latent representations z_i^d and the cluster prototypes c_j^d are L_2-normalized to lie on the unit hypersphere, minimizing the squared Euclidean distance is mathematically equivalent to maximizing the cosine similarity:

$$\|z_i^d - c_j^d\|_2^2 = \|z_i^d\|_2^2 - 2(z_i^d \cdot c_j^d) + \|c_j^d\|_2^2 = 2 - 2(z_i^d \cdot c_j^d).$$

Thus, under unit-norm constraints, the assignment step is equivalent to cosine-similarity based clustering. For initialization, we use the k-means++ method, which samples initial prototypes with probability proportional to their squared distance from existing prototypes, thereby reducing sensitivity to poor initialization.

Following the clustering step, DipContrast estimates a prototype-specific concentration parameter for each cluster. This design is motivated by a heteroscedastic prototype-likelihood perspective, in which each prototype is interpreted not merely as a representative center, but as the center of a local latent distribution with its own cluster-specific spread. In this context, the term "heteroscedastic" signifies that different prototypes may exhibit distinct variances within the latent space, as opposed to assuming a fixed, homogeneous variance profile. From this viewpoint, the prototype posterior is the conditional probability that a latent representation z_i^d is associated with prototype c_j^d. In other words, the prototype posterior measures how likely prototype c_j^d is to explain the embedding z_i^d:

$$p(c_j^d \mid z_i^d) \propto \exp\left(-\frac{\|z_i^d - c_j^d\|_2^2}{2\sigma_j^2}\right), \tag{4.7}$$

where σ_j^2 denotes the prototype-specific variance. Given that the representations and prototypes are L_2-normalized, this formulation naturally translates into a softmax distribution whose logits are scaled by a quantity proportional to σ_j^2, thereby acting as an adaptive temperature. Consequently, dense clusters generate highly confident assignments, while more dispersed clusters result in soft distributions.

To quantify this cluster-specific spread, DipContrast employs a scalar measure of cluster dispersion. For a feature vector $z_i^d \in \mathbb{R}^D$, the dispersion of cluster j is defined as:

$$\mathcal{D}_j^d = \sum_{i \in C_j^d} \sum_{m=1}^{D} \sqrt{\sum_{n=1}^{D} (z_{i,n}^d - c_{j,m}^d)^2}. \tag{4.8}$$

Equivalently, this may be written as:

$$\mathcal{D}_j^d = \sum_{i \in C_j^d} \sum_{m=1}^{D} \|z_i^d - c_{j,m}^d \mathbf{1}\|_2, \tag{4.9}$$

where $\mathbf{1} \in \mathbb{R}^D$ denotes the all-ones vector and $c_{j,m}^d$ is the m-th coordinate of the prototype c_j^d. The corresponding concentration parameter is then given by:

$$\phi_j^d = \frac{\mathcal{D}_j^d}{|C_j^d| \log(|C_j^d| + \alpha)}, \qquad |C_j^d| > 0, \tag{4.10}$$

where $\alpha > 0$ is a smoothing constant and a hyperparameter. If a cluster is empty, its concentration is assigned the current mean of the domain-specific concentration vector.

Furthermore, this serves as a prototype-specific temperature estimator. The numerator $\mathcal{D}_j^d$ acts as a scalar measure of cluster dispersion, increasing with the variability of the samples assigned to cluster j. The denominator normalizes this quantity by the cluster size and further regularizes the estimate through the logarithmic term, thereby preventing small clusters from producing disproportionately large concentration values. Consequently, more compact clusters yield smaller concentration values and thus more confident assignments, whereas more dispersed clusters yield larger concentration values and therefore softer assignment distributions.

4.2.2. M-step: Prototype-Guided Encoder Optimization

During the *M-step*, DipContrast optimizes the encoder by applying two complementary losses to mini-batches of original and augmented representations across both the temporal and frequency domains. First, an instance-level contrastive loss (Equation 4.2) is applied to preserve consistency between paired augmented views and maintain local discriminability in the latent space. Second, Dip-Guided Prototypical Loss (DipProtoLoss) (Equation 4.14) is applied to align the learned representations with the corresponding prototypes in each domain. In practice, both losses are evaluated concurrently on the mini-batch representations, combined into a unified training objective, and minimized via backpropagation to update the encoder parameters. Throughout this optimization stage, the prototypes c_j^d, cluster assignments y_i^d, concentration values ϕ_j^d, and normalized Dip-p-value matrices (see subsection 4.3.3) computed in the *E-step* are used to guide the optimization process. DipProtoLoss is described in section 4.4, while the interplay between the instance-level objective and the prototypical/clustering objective is introduced separately in section 4.5.

In summary, the main training stage alternates between two coupled operations until convergence. The *E-step* computes domain-specific prototypes and their concentrations from the current embedding space, whereas the *M-step* trains the encoder by minimizing the composite loss objective so that the latent feature space progressively organizes into compact clusters around the corresponding prototypes.

Once the iterative *E-step* and *M-step* optimization converges, domain-specific prototype assignments in both the time and frequency latent spaces can be obtained. However, the final discrete cluster labels are derived from the time domain representations. This follows the rationale that the time domain serves as the primary clustering space, while the frequency domain provides complementary structural guidance during training. Accordingly, each sample is assigned to the cluster of its nearest time domain prototype in the final latent space.

4.3. Non-Parametric Evolution

A key feature of DipContrast is its non-parametric clustering mechanism, inspired by DipDECK [54]. Initially, the model is configured with an identically overestimated number of clusters for both domains, such that $k_t = k_f = k_{\text{init}} \gg k$, partitioning each latent space into fine-grained micro-clusters. The subsequent cluster merging occurs primarily during the *E-step*. Unlike DipDECK, DipContrast applies this process independently across the time and frequency domains. Consequently, two distinct prototype sets evolve in parallel, allowing the number of clusters in each domain to decrease at independent rates.

4.3.1. Projection-Based Dip Evaluation

To generate the one-dimensional data required by the Dip-test, DipContrast evaluates each candidate cluster pair by projecting their latent representations onto a single axis. For a given domain $d \in \{t, f\}$, let $\{c_1^d, \ldots, c_{k_d}^d\}$ denote the current prototypes. Let C_a^d and C_b^d denote two candidate clusters, and define $\mathcal{I}_a^d$ and $\mathcal{I}_b^d$ as the sets of indices for the samples assigned to them. Following the DipDECK intuition, this projection axis is chosen as the line connecting the two prototypes, c_a^d and c_b^d. Specifically, we define the connecting vector:

$$u_{ab}^d = c_a^d - c_b^d.$$

All assigned samples are then projected onto this line:

$$\Pi_{ab}^d = \left\{ (z_i^d)^\top u_{ab}^d \ : \ i \in \mathcal{I}_a^d \cup \mathcal{I}_b^d \right\}.$$

The Dip-test is subsequently applied to this projected distribution, Π_{ab}^d. Intuitively, if the projection is unimodal, the two prototypes likely describe the same underlying semantic structure and should be considered merge candidates. Conversely, if the distribution is clearly multimodal, the corresponding clusters are structurally separated and must remain distinct.

4.3.2. Handling Imbalanced Clusters

A practical limitation of projection-based Dip-test arises when candidate clusters differ substantially in size. As noted in DipDECK, large imbalances between clusters can bias the Dip-test toward high Dip-p-value and cause incorrect merges of clusters with a large spatial distance between them. To counteract this, DipContrast adopts the balancing principle introduced in DipDECK.

Let C_a^d and C_b^d denote two candidate clusters in domain d, with cardinalities $n_a^d = |C_a^d|$ and $n_b^d = |C_b^d|$. If one cluster is larger than the other by more than a defined factor β, a second Dip-test is performed on a balanced subset. Suppose that:

$$n_a^d > \beta n_b^d.$$

In this case, only the $\lfloor \beta n_b^d \rfloor$ samples from the larger cluster C_a^d that are closest to the smaller-cluster prototype c_b^d are retained. The second Dip-p-value is then computed using all points from the smaller cluster together with this selected subset from the larger cluster. Let $p_{ab}^{d,\text{full}}$ denote the Dip-p-value obtained from the full-cluster projection and let $p_{ab}^{d,\text{bal}}$ denote the value obtained from the balanced subset. The final Dip-p-value is defined as:

$$p_{ab}^d = \min\left(p_{ab}^{d,\text{full}}, p_{ab}^{d,\text{bal}}\right).$$

Thus, the final score is determined by the more conservative of the two evaluations, reducing the risk that a small but distinct cluster is absorbed into a much larger cluster. This balancing procedure is applied symmetrically: if instead $n_b^d > \beta n_a^d$, then the same operation is performed with the roles of the two clusters reversed.

4.3.3. Dip-p-Value Matrices and Their Normalization

After all pairwise prototype comparisons have been evaluated, DipContrast constructs a symmetric Dip-p-value matrix separately for each domain. For the temporal domain, this yields:

$$P^t = [p_{ab}^t] \in [0,1]^{k_t \times k_t},$$

and analogously for the frequency domain:

$$P^f = [p_{ab}^f] \in [0,1]^{k_f \times k_f}.$$

Each off-diagonal entry p_{ab}^d stores the Dip-p-value associated with prototype pair c_a^d and c_b^d, whereas the diagonal entries are fixed to one:

$$p_{aa}^d = 1.$$

Since the Dip-p-value quantifies the likelihood of unimodality, large values indicate that the two candidate micro-clusters are likely to belong to a shared latent structure, while small values indicate clear separation. Maintaining both P^t and P^f is necessary because DipContrast learns and clusters time-series data in two complementary latent spaces.

The Dip-p-value matrices are subsequently normalized row-wise. For each domain $d \in \{t, f\}$, the normalized matrix elements are defined as:

$$\hat{p}_{ab}^d = \frac{p_{ab}^d}{\sum_{\ell=1}^{k_d} p_{a\ell}^d}.$$

This normalization converts each row into a valid distribution over prototypes and places the entries on a common relative scale. Following the same intuition as in DipDECK, the row-wise normalization yields a weighting over prototypes. In the present setting, this is particularly useful because the normalized Dip-p-value matrices are later used in the prototype-guided optimization stage as soft target distributions.

4.3.4. Cluster Merging Process

DipContrast executes the cluster merging process independently across the time and frequency domains. In a given domain d, merging is triggered only when three conditions are satisfied: (i) the current number of clusters is greater than one, (ii) the maximum off-diagonal Dip-p-value exceeds a predefined threshold τ_{dip}, and (iii) a defined merge interval is reached. This scheduled merging prevents prototypes from being updated too aggressively and allows the latent space to stabilize between successive merge operations. Formally, for domain d, a merge is initiated when

$$\max_{a \neq b} p^d_{ab} \geq \tau_{\text{dip}},$$

subject to the merge schedule imposed during training. Once the merge condition is met, the pair of prototypes with the highest off-diagonal Dip-p-value is selected:

$$(a^\star, b^\star) = \arg\max_{a \neq b} p^d_{ab}.$$

The labels of all samples assigned to these two clusters are then merged, and a new prototype is formed as the size-weighted average of the two original prototypes:

$$c^d_{\text{new}} = \frac{|C^d_{a^\star}|\, c^d_{a^\star} + |C^d_{b^\star}|\, c^d_{b^\star}}{|C^d_{a^\star}| + |C^d_{b^\star}|}. \tag{4.11}$$

After each merge, the number of clusters in the relevant domain is reduced by one, and the corresponding concentration values and Dip-p-value matrices are re-estimated under the updated prototype configuration. This ensures that every merge decision is followed by a fresh assessment of the resulting prototype geometry. The temporal and frequency domains undergo non-parametric evolution independently, maintaining separate cluster counts, prototype sets, and merge trajectories. Although the two domains are optimized jointly at the representation-learning level, their prototype refinement remains domain-specific, allowing each latent space to adapt its own unique characteristics.

4.4. Dip-Guided Prototypical Loss

To incorporate the non-parametric relations discovered during the *E-step* into representation learning, DipContrast employs Dip-Guided Prototypical Loss (DipProtoLoss). Inspired by PCL, this objective replaces one-hot prototype targets with soft target distributions derived from the normalized Dip p-value matrix. As a result, each sample is encouraged not only to align with its assigned prototype, but also to respect the structural affinities among neighboring prototypes identified by the Dip-test.

Let z^d_i be the feature embedding for instance i in domain $d \in \{t, f\}$, and let s^d_i denote the index of its assigned cluster. We define $\hat{p}^d_{s^d_i, j}$ as the normalized Dip p-value matrix entry representing the probabilistic relationship between this assigned cluster s^d_i and any other target cluster j. This row serves as the target distribution. The loss function is a

soft cross-entropy minimization between this target distribution and the model's predicted probability distribution over the clusters. Thus, the Dip-guided prototypical loss for the representation z_i^d is formally defined as:

$$\mathcal{L}_{z_i^d}^{\text{DipProto}} = -\sum_{j=1}^{k_d} \hat{p}_{s_i^d,j}^d \log \left(\frac{\exp\Big(\text{sim}(z_i^d, c_j^d)/\phi_j^d\Big)}{\sum_{m=1}^{k_d} \exp\big(\text{sim}(z_i^d, c_m^d)/\phi_m^d\big)} \right), \tag{4.12}$$

where c_j^d is the prototype for cluster j, and ϕ_j^d denotes the estimated concentration parameter for that cluster. Additionally, $\text{sim}(\cdot,\cdot)$ computes the cosine similarity between the L_2-normalized vectors. Similarly, for the augmented representation $\tilde{z}_i^d$, with assigned prototype index $\tilde{s}_i^d$, the corresponding loss is computed as:

$$\mathcal{L}_{\tilde{z}_i^d}^{\text{DipProto}} = -\sum_{j=1}^{k_d} \hat{p}_{\tilde{s}_i^d,j}^d \log \left(\frac{\exp\Big(\text{sim}(\tilde{z}_i^d, c_j^d)/\phi_j^d\Big)}{\sum_{m=1}^{k_d} \exp\big(\text{sim}(\tilde{z}_i^d, c_m^d)/\phi_m^d\big)} \right). \tag{4.13}$$

The total Dip-guided prototypical loss for a given domain is then obtained by summing the average losses of the original samples and their augmented views:

$$\mathcal{L}^{\text{DipProto}}(Z^d, \tilde{Z}^d) = \frac{1}{n} \sum_{i=1}^{n} \left(\mathcal{L}_{z_i^d}^{\text{DipProto}} + \mathcal{L}_{\tilde{z}_i^d}^{\text{DipProto}} \right). \tag{4.14}$$

The loss is applied independently in the temporal and frequency domains, and plays a twofold role in shaping the latent space. First, it retains the prototype-based supervisory effect of PCL by encouraging embeddings to align tightly with their assigned prototypes. Second, by utilizing the normalized Dip-p-value matrix as a soft target distribution, the semantic cluster relations discovered during the *E-step* are integrated directly into the optimization process. Consequently, rather than collapsing into isolated clusters, the latent embeddings are optimized to preserve the semantic grouping of the data.

4.5. Cross-Domain Consensus

The objective of DipContrast is to learn latent representations that remain discriminative within each domain while preserving semantic consistency across the time and frequency views of the same time-series. Consequently, the training framework jointly optimizes instance-level invariance, semantic groupings, and cross-domain agreement. For each domain, the optimization objective combines the instance-level contrastive loss (Equation 4.2) and the Dip-Guided Prototypical Loss (Equation 4.14) into a composite loss, with γ regulating the influence of the Dip-guided prototypical component:

$$\mathcal{L}_t^{\text{training}} = \mathcal{L}^{\text{Ins}}(Z^t, \tilde{Z}^t) + \gamma \mathcal{L}^{\text{DipProto}}(Z^t, \tilde{Z}^t), \tag{4.15}$$

$$\mathcal{L}_f^{\text{training}} = \mathcal{L}^{\text{Ins}}(Z^f, \tilde{Z}^f) + \gamma \mathcal{L}^{\text{DipProto}}(Z^f, \tilde{Z}^f). \tag{4.16}$$

Beyond these intra-domain objectives, DipContrast introduces a cross-domain consensus term to align the temporal and frequency views of the same sample. Importantly, only the instance-level contrastive loss is applied across domains, while DipProtoLoss remains domain-specific. Since the time and frequency domains maintain independently evolving prototype sets, a one-to-one mapping between cross-domain cluster indices cannot be established reliably. Furthermore, the cross-domain constraint is applied solely to the augmented representations. As outlined in CDCC, applying cross-domain constraints directly to the original data results in model overfitting. The cross-domain consensus loss is thus defined as:

$$\mathcal{L}_{tf}^{\text{training}} = \mathcal{L}^{\text{Ins}}(\tilde{Z}^t, \tilde{Z}^f). \tag{4.17}$$

The overall objective minimized during the training step is then defined as:

$$\mathcal{L}^{\text{training}} = \lambda(\mathcal{L}_t^{\text{training}} + \mathcal{L}_f^{\text{training}}) + (1-\lambda)\mathcal{L}_{tf}^{\text{training}}. \tag{4.18}$$

where $\lambda \in [0, 1]$ balances domain-specific structural refinement against cross-domain alignment. Overall, this unified objective allows the time and frequency domains to develop their own prototype geometries while preserving semantic agreement between the two views for each individual sample.

Input: Dataset $\mathcal{X} = \{(x_i^t, x_i^f)\}_{i=1}^N$, initial cluster count k_{init}, warm-up epochs E_{warm}, total epochs E_{total}, merge interval T_{merge}, Dip threshold τ_{dip}, λ, γ, α, β

Output: Trained encoder parameters θ

Initialization: Initialize the dual-domain encoder and projection heads with parameters θ; set $k_t \leftarrow k_{\text{init}}$, $k_f \leftarrow k_{\text{init}}$;

for $e \leftarrow 1$ **to** E_{warm} **do**
  foreach *mini-batch* $\mathcal{B} \subset \mathcal{X}$ **do**
    Generate augmented views $\tilde{X}_{\mathcal{B}}^t$ and $\tilde{X}_{\mathcal{B}}^f$;
    Compute embeddings $Z_{\mathcal{B}}^t$, $\tilde{Z}_{\mathcal{B}}^t$, $Z_{\mathcal{B}}^f$, and $\tilde{Z}_{\mathcal{B}}^f$;
    $\mathcal{L}_t^{\text{Ins}} \leftarrow \mathcal{L}^{\text{Ins}}(Z_{\mathcal{B}}^t, \tilde{Z}_{\mathcal{B}}^t)$; $\mathcal{L}_f^{\text{Ins}} \leftarrow \mathcal{L}^{\text{Ins}}(Z_{\mathcal{B}}^f, \tilde{Z}_{\mathcal{B}}^f)$; $\mathcal{L}_{tf}^{\text{Ins}} \leftarrow \mathcal{L}^{\text{Ins}}(\tilde{Z}_{\mathcal{B}}^t, \tilde{Z}_{\mathcal{B}}^f)$;
    $\mathcal{L}^{\text{warm-up}} \leftarrow \lambda\big(\mathcal{L}_t^{\text{Ins}} + \mathcal{L}_f^{\text{Ins}}\big) + (1-\lambda)\mathcal{L}_{tf}^{\text{Ins}}$;
    Update $\theta \leftarrow \text{Optimizer}(\nabla \mathcal{L}^{\text{warm-up}})$;

for $e \leftarrow E_{\text{warm}}$ **to** E_{total} **do**
  `// E-step: Prototype, Concentration Est., Non-Param. Evolution`
  foreach $d \in \{t, f\}$ **do**
    Extract all latent representations Z^d using θ;
    Run k-means$(Z^d, k_d) \rightarrow$ prototypes $C^d = \{c_1^d, \ldots, c_{k_d}^d\}$ and assignments y^d;
    Estimate concentration vector $\phi^d = \{\phi_1^d, \ldots, \phi_{k_d}^d\}$ from (Z^d, C^d, y^d, α);
    Compute Dip-p-value matrix $P^d = [p_{ab}^d]$ using (Z^d, C^d, y^d, β);
    if $(k_d > 1)$ ***and*** $(e \bmod T_{\text{merge}} = 0)$ ***and*** $\big(\max_{a \neq b} p_{ab}^d \geq \tau_{\text{dip}}\big)$ **then**
      Select pair $(a^\star, b^\star) = \arg\max_{a \neq b} p_{ab}^d$;
      Merge clusters $C_{a^\star}^d$ and $C_{b^\star}^d$;
      Update $k_d \leftarrow k_d - 1$;
      Recompute prototypes C^d, assignments y^d, concentrations ϕ^d, and Dip matrix P^d;
    Normalize Dip matrix $\hat{P}^d \leftarrow \text{RowNormalize}(P^d)$;
  `// M-step: Prototype-Guided Representation Learning`
  foreach *mini-batch* $\mathcal{B} \subset \mathcal{X}$ **do**
    Generate augmented views $\tilde{X}_{\mathcal{B}}^t$ and $\tilde{X}_{\mathcal{B}}^f$;
    Compute embeddings $Z_{\mathcal{B}}^t$, $\tilde{Z}_{\mathcal{B}}^t$, $Z_{\mathcal{B}}^f$, and $\tilde{Z}_{\mathcal{B}}^f$;
    $\mathcal{L}_t^{\text{training}} \leftarrow \mathcal{L}^{\text{Ins}}(Z_{\mathcal{B}}^t, \tilde{Z}_{\mathcal{B}}^t) + \gamma\, \mathcal{L}^{\text{DipProto}}(Z_{\mathcal{B}}^t, \tilde{Z}_{\mathcal{B}}^t; C^t, \phi^t, \hat{P}^t)$;
    $\mathcal{L}_f^{\text{training}} \leftarrow \mathcal{L}^{\text{Ins}}(Z_{\mathcal{B}}^f, \tilde{Z}_{\mathcal{B}}^f) + \gamma\, \mathcal{L}^{\text{DipProto}}(Z_{\mathcal{B}}^f, \tilde{Z}_{\mathcal{B}}^f; C^f, \phi^f, \hat{P}^f)$;
    $\mathcal{L}_{tf}^{\text{training}} \leftarrow \mathcal{L}^{\text{Ins}}(\tilde{Z}_{\mathcal{B}}^t, \tilde{Z}_{\mathcal{B}}^f)$;
    $\mathcal{L}^{\text{training}} \leftarrow \lambda\big(\mathcal{L}_t^{\text{training}} + \mathcal{L}_f^{\text{training}}\big) + (1-\lambda)\mathcal{L}_{tf}^{\text{training}}$;
    Update $\theta \leftarrow \text{Optimizer}(\nabla \mathcal{L}^{\text{training}})$;

Algorithm 1: DipContrast: A Contrastive Learning Framework for Non-Parametric Deep Time-Series Clustering

5. Experiments

5.1. Characteristics of the Datasets

We evaluate DipContrast on time-series datasets both with and without ground-truth labels. Labeled datasets are used to assess clustering accuracy and validate the estimated number of clusters. On the other hand, unlabeled datasets demonstrate the algorithm's practical applicability in fully unsupervised, real-world scenarios.

5.1.1. UCR Datasets

The UCR Time Series Classification Archive [55] serves as our primary benchmark for labeled data. We utilize a subset of 40 representative datasets from this archive to ensure a thorough evaluation across various domains, sequence lengths, and time-dependent variations. The selected datasets and their characteristics are provided in Table 5.1.

Dataset Name	Type	Train	Test	Class	Length
ACSF1	Device	100	100	10	1460
Adiac	Image	390	391	37	176
ArrowHead	Image	36	175	3	251
Beef	Spectro	30	30	5	470
CBF	Simulated	30	900	3	128
Car	Sensor	60	60	4	577
CricketX	Motion	390	390	12	300
CricketY	Motion	390	390	12	300
CricketZ	Motion	390	390	12	300
DiatomSizeReduction	Image	16	306	4	345
DistalPhalanxOutlineAgeGroup	Image	400	139	3	80
DistalPhalanxTW	Image	400	139	6	80
ECG200	ECG	100	100	2	96
ECGFiveDays	ECG	23	861	2	136
EOGVerticalSignal	EOG	362	362	12	1250
FaceFour	Image	24	88	4	350
FiftyWords	Image	450	455	50	270
Fish	Image	175	175	7	463
Fungi	HRM	18	186	18	201
GunPointMaleVersusFemale	Motion	135	316	2	150

Continued on next page

Table 5.1 – continued from previous page

Name	Type	Train	Test	Class	Sequence Length
GunPointOldVersusYoung	Motion	136	315	2	150
HouseTwenty	Device	40	119	2	2000
LargeKitchenAppliances	Device	375	375	3	720
MiddlePhalanxOutlineAgeGroup	Image	400	154	3	80
MiddlePhalanxTW	Image	399	154	6	80
OSULeaf	Image	200	242	6	427
PigAirwayPressure	Hemodynamics	104	208	52	2000
PigArtPressure	Hemodynamics	104	208	52	2000
PigCVP	Hemodynamics	104	208	52	2000
Plane	Sensor	105	105	7	144
PowerCons	Power	180	180	2	144
ProximalPhalanxTW	Image	400	205	6	80
SemgHandMovementCh2	Spectrum	450	450	6	1500
ShapeletSim	Simulated	20	180	2	500
ShapesAll	Image	600	600	60	512
SwedishLeaf	Image	500	625	15	128
SyntheticControl	Simulated	300	300	6	60
ToeSegmentation1	Motion	40	228	2	277
Trace	Sensor	100	100	4	275
WordSynonyms	Image	267	638	25	270

Table 5.1.: Characteristics of the selected UCR Archive Datasets.

5.1.2. Smart meter dataset

The smart meter dataset consists of energy usage data from more than 250,000 consumers in Austria and contains no ground-truth labels. The data follows a specific structure: it includes an ID identifying the smart meter device, a real number indicating energy consumption in kilowatt-hours (kWh), two timestamps marking the start and end of the measurement period, and the type of measurement. The measurement types define data granularity and are categorized into two types, namely **IMS** and **IME**. IMS stands for *Intelligentes Messgerät in der Standardkonfiguration* (translation: smart meter in standard configuration) with daily granularity, and IME stands for *Intelligentes Messgerät in der erweiterten Konfiguration* (translation: smart meter in extended configuration) with 15-minute granularity. Due to legal regulations in Austria, data is transmitted at 15-minute intervals only for customers who have explicitly opted in for the IME option. In practice, the number of consumers who explicitly opt in remains comparatively small. Thus, to generalize across the dataset, we aggregate the IME instances to a daily granularity for our evaluation. The dataset spans one year, from January 1, 2023, to December 31, 2023. This is a proprietary dataset and is not publicly available.

5.2. Evaluation Metrics

Evaluating the quality of clustering results is a critical aspect of time-series clustering, as it directly influences the reliability, interpretability, and practical applicability of the discovered patterns. Evaluation methodologies are broadly categorized into *external* and *internal* indices [56]. External indices are used when ground-truth labels are available. These metrics compare the output of a clustering algorithm with an existing "true" classification to assess how accurately the algorithm has performed. In contrast, internal indices assess the quality of clustering based on properties inherent to the data and the clustering structure, without requiring external labels. These indices rely on evaluating how well data points within the same cluster resemble each other (*cohesion*) and how distinct different clusters are (*separation*). Each method has its advantages and drawbacks. We choose the following metrics based on their usability and viability in our context for evaluating clustering results:

5.2.1. External metrics

Normalized Mutual Information (NMI)

NMI measures the agreement between the predicted clusters and the true labels by quantifying the amount of shared information between them, normalized to ensure comparability across different clustering assignments. NMI is based on the concept of mutual information from information theory, which captures how much knowing one variable (e.g., predicted cluster label) reduces uncertainty about another (e.g., true label). However, since mutual information favors partitions with many clusters, it is typically normalized based on the entropies of the predicted and true assignments.

$$\mathrm{NMI}(Y, C) = \frac{2 \times I(Y; C)}{[H(Y) + H(C)]},$$

where:

- $I(Y; C)$: Mutual Information between Y and C.
- $H(\cdot)$: Entropy.

The NMI score lies between 0 and 1, with a value of 1 representing complete agreement between the two clustering assignments, while a value of 0 indicates that the clustering assignments share no mutual information.

Rand Index (RI)

The Rand Index evaluates the similarity between the predicted clustering and the ground-truth labels by considering all pairs of samples and assessing whether they are grouped consistently. Essentially, it translates the clustering evaluation into a pairwise binary classification problem by calculating the ratio of agreeing pairs to the total number of possible pairs. Agreeing pairs consist of elements that are correctly assigned to the same cluster (True Positives) or correctly assigned to different clusters (True Negatives).

$$\mathrm{RI} = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}},$$

where:

- a: Number of pairs of elements that are in the same cluster in both the true labels and the predicted clusters.
- b: Number of pairs of elements that are in different clusters in both the true labels and the predicted clusters.
- c: Number of pairs of elements that are in the same true cluster but different predicted clusters.
- d: Number of pairs of elements that are in different true clusters but the same predicted cluster.
- n: Total number of data points.

The RI score ranges from 0 to 1, with a value of 1 indicating perfect agreement between the predicted clusters and the ground truth. A score of 0 indicates that the predicted clusters and the ground truth do not agree on the placement of any pair of points.

5.2.2. Internal metrics

Silhouette Score

The Silhouette score is a widely used internal cluster evaluation metric that measures how similar a data point is to its own cluster (cohesion) compared to other clusters (separation). For each data point i, the individual silhouette coefficient s_i is defined using two values:

- a_i: the mean intra-cluster distance, i.e., the average distance between i and all other points in the same cluster.
- b_i: the minimum average distance from data point i to data points in a different cluster.

The coefficient s_i is then calculated as:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}.$$

To evaluate the clustering result as a whole, the overall silhouette score S is computed by taking the average of the silhouette coefficients for all N data points in the dataset:

$$\mathrm{S} = \frac{1}{N}\sum_{i=1}^{N} s_i.$$

The overall silhouette score S ranges from -1 to 1. A score approaching 1 indicates that the clusters are highly cohesive and well-separated, while a score near -1 suggests overlapping clusters or poor grouping. Consequently, a higher overall score S signifies a more robust and better-defined clustering structure.

Davies-Bouldin (DB) Score

The Davies-Bouldin score is a prominent internal clustering evaluation metric that assesses clustering quality by simultaneously considering intra-cluster dispersion and inter-cluster separation. It provides a balanced measure by evaluating how compact each cluster is and how distinct it is from others. The dispersion S_i for a cluster C_i is computed as the average distance between the data points in the cluster and its centroid c_i, and the separation between clusters C_i and C_j is determined by the distance between their respective centroids. For a clustering with K total clusters, the DB score is defined as:

$$\text{DB} = \frac{1}{K} \sum_{i=1}^{K} \max_{\substack{j=1,\dots,K \\ j \neq i}} \left(\frac{S_i + S_j}{M_{ij}} \right),$$

where

$$S_i = \frac{1}{|C_i|} \sum_{x \in C_i} d(x, c_i), \quad \text{and} \quad M_{ij} = d(c_i, c_j).$$

A lower DB score signifies better clustering, characterized by tightly cohesive clusters that are clearly separated from one another.

5.3. Experimental Setup

To evaluate the effectiveness of DipContrast, we compare it with four partition-based non-parametric clustering algorithms, serving as our baselines: X-means [32], G-means [33], PG-means [34], and Dip-means [35]. For all baseline methods, we use the implementations provided by the `clustpy` library [57]. Each baseline is evaluated under two settings: (i) on z-normalized time-series data, and (ii) on latent representations obtained after the warm-up phase of DipContrast. The latter setting isolates the contribution of the learned representation space, independent of the proposed clustering mechanism. All experiments were implemented in `PyTorch` [1] and conducted on a computing environment featuring dual Intel Xeon Gold 6130 CPUs (32 cores total), 320 GB RAM, and two NVIDIA RTX 4090 GPUs (24 GB VRAM each).

For UCR datasets, the train and test sets are combined for evaluation. We employ an adaptive parameter configuration strategy, sampling hyperparameters from a predefined search space for each dataset. These include time-series augmentation parameters (e.g., σ_{jitter}, σ_{scaling}, W), optimization parameters (learning rate, weight decay), architectural choices (encoder dimensions), and clustering-specific parameters (e.g., γ, λ, τ_{dip}).

Batch sizes are selected adaptively based on the size of the dataset. For UCR datasets, the batch size search space spans up to 40% of the total dataset size, with candidate

[1] `https://www.pytorch.org/`

values restricted to powers of two and capped at 256. This balances computational efficiency with the large batch size requirements of a SimCLR-based contrastive learning architecture. Conversely, for the significantly larger smart meter dataset, the batch size search space is defined as $\{1024, 2048, 4096\}$.

Furthermore, all algorithms expect an initial overestimation of k, denoted as k_{init}. For UCR datasets, we overestimate the maximum number of clusters using $k_{\text{init}} = 3C + \lfloor \frac{C}{3} \rfloor$, where C denotes the ground-truth number of classes. For the smart meter dataset, k_{init} is empirically set to 100. To ensure fairness, all algorithms are provided with the same k_{init} for each dataset. DipContrast is trained for a total of 300 epochs across all datasets, with the warm-up period optimized from a search space of $\{20, 30, 40\}$ epochs. Additional hyperparameters and their associated search spaces are detailed in Table A.1. The results reported in Table 5.2 and Table 5.3 are based on the best-performing hyperparameter configuration for each dataset.

5.4. Results

5.4.1. UCR Datasets

Method	**ACSF1**			**Adiac**			**ArrowHead**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	10	-	-	37	-	-	3	-	-
X-means	7	0.48	0.75	70	**0.62**	0.96	10	**0.32**	0.66
G-means	27	**0.58**	**0.87**	25	0.56	0.92	**3**	0.31	0.58
PG-means	**10**	0.37	0.55	**28**	0.25	0.60	11	0.06	0.38
Dip-means	1	0.00	0.10	1	0.00	0.03	1	0.00	0.33
Warm-up + X-means	32	0.52	**0.87**	123	0.51	**0.97**	10	0.24	**0.68**
Warm-up + G-means	**10**	0.49	**0.87**	16	0.37	0.88	5	0.24	**0.68**
Warm-up + PG-means	17	0.49	0.81	13	0.35	0.78	**3**	0.22	0.48
Warm-up + Dip-means	2	0.22	0.52	6	0.31	0.81	**3**	0.28	0.66
DipContrast (ours)	8	0.46	0.82	69	**0.62**	**0.97**	2	0.27	0.58

Method	**Beef**			**Car**			**CBF**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	5	-	-	4	-	-	3	-	-
X-means	14	0.38	0.76	13	0.33	0.74	10	0.52	0.74
G-means	3	0.24	0.61	1	0.00	0.24	10	0.50	0.74
PG-means	2	0.00	0.19	1	0.00	0.24	1	0.00	0.33
Dip-means	1	0.00	0.19	1	0.00	0.24	1	0.00	0.33
Warm-up + X-means	16	**0.41**	**0.78**	13	0.25	0.74	10	0.63	0.78
Warm-up + G-means	**4**	0.23	0.64	**6**	0.24	0.70	10	0.61	0.80
Warm-up + PG-means	2	0.03	0.21	7	0.13	0.63	11	0.42	0.63
Warm-up + Dip-means	2	0.12	0.53	1	0.00	0.24	9	0.65	0.79
DipContrast (ours)	12	0.38	0.75	7	**0.42**	**0.76**	**3**	**1.00**	**1.00**

Continued on next page

Table 5.2 – continued from previous page

Method	CricketX			CricketY			CricketZ		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	12	-	-	12	-	-	12	-	-
X-means	29	0.34	0.90	32	0.35	0.90	29	0.33	0.90
G-means	38	0.38	0.90	17	0.33	0.88	40	0.40	0.90
PG-means	**4**	0.02	0.45	**8**	0.05	0.32	22	0.14	0.68
Dip-means	1	0.00	0.08	2	0.14	0.53	1	0.00	0.08
Warm-up + X-means	40	0.39	0.91	40	0.39	**0.91**	40	**0.47**	**0.91**
Warm-up + G-means	33	0.39	0.91	17	0.36	0.89	27	0.45	**0.91**
Warm-up + PG-means	1	0.00	0.08	5	0.01	0.09	1	0.00	0.08
Warm-up + Dip-means	1	0.00	0.08	2	0.05	0.49	1	0.00	0.08
DipContrast (ours)	27	**0.50**	**0.92**	19	**0.44**	0.90	**20**	0.45	0.90

Method	DiatomSize Reduction			DistalPhalanx OutlineAgeGroup			DistalPhalanx TW		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	4	-	-	3	-	-	6	-	-
X-means	13	0.77	0.88	10	0.28	0.60	20	0.40	0.73
G-means	7	**0.94**	**0.99**	10	0.28	0.60	10	0.42	0.74
PG-means	10	0.32	0.60	11	0.01	0.47	21	0.29	0.73
Dip-means	3	0.82	0.92	**2**	0.35	0.70	2	0.57	0.79
Warm-up + X-means	13	0.53	0.79	10	0.28	0.60	20	0.40	0.73
Warm-up + G-means	8	0.64	0.84	10	0.28	0.60	8	0.48	0.77
Warm-up + PG-means	14	0.06	0.30	11	0.14	0.59	**5**	0.41	0.69
Warm-up + Dip-means	**4**	0.66	0.84	**2**	0.36	**0.71**	2	0.56	0.79
DipContrast (ours)	3	0.81	0.92	**2**	**0.38**	**0.71**	3	**0.64**	**0.83**

Method	ECG200			ECGFiveDays			EOGVertical Signal		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	2	-	-	2	-	-	12	-	-
X-means	6	0.25	0.59	6	0.00	0.50	40	**0.32**	0.89
G-means	5	0.19	0.57	6	0.01	0.50	16	0.28	0.86
PG-means	**1**	0.00	0.55	4	0.01	0.50	21	0.10	0.51
Dip-means	**1**	0.00	0.55	**3**	0.00	0.50	1	0.00	0.08
Warm-up + X-means	6	0.21	0.52	6	0.10	0.55	40	0.29	**0.90**
Warm-up + G-means	5	0.22	0.56	6	0.10	0.55	**9**	0.19	0.81
Warm-up + PG-means	4	0.15	0.56	7	0.08	0.55	19	0.13	0.65
Warm-up + Dip-means	4	0.24	0.56	7	0.15	0.57	2	0.12	0.52
DipContrast (ours)	**3**	**0.34**	**0.69**	**3**	**0.59**	**0.80**	5	0.17	0.76

Method	FaceFour			FiftyWords			Fish		

Continued on next page

Table 5.2 – continued from previous page

	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	4	-	-	50	-	-	7	-	-
X-means	8	0.45	0.78	**62**	0.64	0.95	16	0.40	0.84
G-means	**4**	0.45	0.72	24	0.55	0.94	11	0.39	0.82
PG-means	1	0.00	0.25	1	0.00	0.04	1	0.00	0.14
Dip-means	1	0.00	0.25	1	0.00	0.04	1	0.00	0.14
Warm-up + X-means	13	0.45	0.78	63	0.62	0.95	23	0.34	**0.85**
Warm-up + G-means	7	0.49	0.77	21	0.52	0.93	**7**	0.36	0.82
Warm-up + PG-means	1	0.00	0.25	1	0.00	0.04	3	0.25	0.44
Warm-up + Dip-means	6	0.49	0.77	1	0.00	0.04	5	0.37	0.79
DipContrast (ours)	5	**0.54**	**0.81**	110	**0.67**	**0.96**	5	**0.43**	0.75

Method	**Fungi**			**GunPoint MaleVersusFemale**			**GunPoint OldVersusYoung**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	18	-	-	2	-	-	2	-	-
X-means	57	0.83	0.96	6	0.37	0.62	6	0.58	0.70
G-means	**14**	0.86	0.94	6	0.37	0.62	6	0.58	0.70
PG-means	1	0.00	0.06	7	0.17	0.52	7	0.10	0.55
Dip-means	1	0.00	0.06	5	0.34	0.57	5	0.66	0.79
Warm-up + X-means	52	0.79	0.95	6	0.33	0.61	6	0.63	0.79
Warm-up + G-means	**14**	0.75	0.92	6	0.33	0.61	6	0.64	0.79
Warm-up + PG-means	8	0.52	0.71	7	0.35	0.59	7	0.34	0.62
Warm-up + Dip-means	9	0.69	0.89	5	0.30	0.57	5	0.66	0.79
DipContrast (ours)	31	**0.87**	**0.97**	**3**	**0.42**	**0.64**	**3**	**0.79**	**0.86**

Method	**HouseTwenty**			**LargeKitchen Appliances**			**MiddlePhalanx OutlineAgeGroup**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	2	-	-	3	-	-	3	-	-
X-means	4	0.10	0.55	10	0.04	0.62	10	0.28	0.66
G-means	5	0.09	0.54	10	0.03	0.62	**3**	0.40	0.73
PG-means	**1**	0.00	0.50	7	0.03	0.49	11	0.01	0.39
Dip-means	**1**	0.00	0.50	1	0.00	0.33	2	**0.46**	0.70
Warm-up + X-means	6	0.08	0.52	10	0.14	**0.66**	10	0.28	0.64
Warm-up + G-means	4	0.08	0.53	10	0.15	**0.66**	8	0.33	**0.75**
Warm-up + PG-means	**1**	0.00	0.50	**3**	0.06	0.48	2	0.00	0.38
Warm-up + Dip-means	**1**	0.00	0.50	4	0.13	0.63	2	**0.46**	0.70
DipContrast (ours)	4	**0.14**	**0.56**	**3**	**0.17**	0.63	2	**0.46**	0.70

Method	**MiddlePhalanx TW**			**OSULeaf**			**PigAirway Pressure**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI

Continued on next page

Table 5.2 – continued from previous page

Ground truth	6	-	-	6	-	-	52	-	-
X-means	20	0.33	0.77	20	0.28	0.81	30	0.50	0.95
G-means	**6**	0.44	**0.84**	9	0.26	0.78	8	0.22	0.85
PG-means	14	0.05	0.38	1	0.00	0.18	5	0.08	0.66
Dip-means	2	**0.55**	0.71	1	0.00	0.18	1	0.00	0.02
Warm-up + X-means	20	0.33	0.77	20	0.36	**0.82**	**64**	0.63	0.97
Warm-up + G-means	4	0.46	0.83	8	0.39	0.81	6	0.29	0.83
Warm-up + PG-means	9	0.22	0.42	1	0.00	0.18	1	0.00	0.02
Warm-up + Dip-means	2	0.54	0.71	4	0.41	0.75	1	0.00	0.02
DipContrast (ours)	2	0.54	0.71	**5**	**0.49**	0.81	165	**0.77**	**0.98**

Method	**PigArt Pressure**			**PigCVP**			**Plane**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	52	-	-	52	-	-	7	-	-
X-means	16	0.52	0.91	**19**	0.48	0.92	23	0.76	0.91
G-means	1	0.00	0.02	15	0.45	0.90	8	0.91	0.95
PG-means	1	0.00	0.02	1	0.00	0.02	11	0.62	0.76
Dip-means	1	0.00	0.02	1	0.00	0.02	6	0.88	0.94
Warm-up + X-means	**62**	0.74	**0.98**	173	0.76	**0.98**	23	0.77	0.92
Warm-up + G-means	10	0.61	0.90	8	0.47	0.87	10	0.90	0.97
Warm-up + PG-means	3	0.00	0.02	1	0.00	0.02	3	0.10	0.22
Warm-up + Dip-means	1	0.00	0.02	1	0.00	0.02	**7**	**0.93**	**0.98**
DipContrast (ours)	147	**0.83**	**0.98**	163	**0.78**	**0.98**	**7**	**0.93**	**0.98**

Method	**PowerCons**			**ProximalPhalanx TW**			**SemgHand MovementCh2**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	2	-	-	6	-	-	6	-	-
X-means	6	0.37	0.64	20	0.41	0.73	12	0.33	0.77
G-means	6	0.44	0.65	**4**	0.56	0.85	20	**0.34**	0.80
PG-means	1	0.00	0.50	21	0.12	0.42	3	0.22	0.62
Dip-means	1	0.00	0.50	2	0.62	0.78	1	0.00	0.17
Warm-up + X-means	6	0.40	0.68	20	0.48	0.74	20	0.24	**0.81**
Warm-up + G-means	6	0.43	0.68	15	0.52	0.77	15	0.25	0.80
Warm-up + PG-means	6	0.13	0.50	21	0.44	0.66	19	0.25	0.76
Warm-up + Dip-means	**2**	**0.58**	**0.84**	3	0.64	0.83	5	0.25	0.73
DipContrast (ours)	6	0.54	0.82	3	**0.69**	**0.88**	**6**	0.25	0.74

Method	**ShapeletSim**			**ShapesAll**			**SwedishLeaf**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	2	-	-	60	-	-	15	-	-
X-means	**2**	0.00	0.50	64	0.70	0.97	50	0.60	0.93
G-means	6	0.01	0.50	80	**0.75**	**0.98**	35	0.62	**0.94**

Continued on next page

Table 5.2 – continued from previous page

PG-means	1	0.00	0.50	110	0.33	0.62	51	0.07	0.14
Dip-means	1	0.00	0.50	2	0.16	0.51	1	0.00	0.07
Warm-up + X-means	6	**0.04**	0.51	126	0.73	**0.98**	50	0.60	**0.94**
Warm-up + G-means	6	**0.04**	0.51	**58**	0.70	0.97	28	0.61	**0.94**
Warm-up + PG-means	4	0.00	0.50	33	0.42	0.77	3	0.01	0.07
Warm-up + Dip-means	1	0.00	0.50	2	0.24	0.48	1	0.00	0.07
DipContrast (ours)	5	**0.04**	**0.52**	144	0.74	**0.98**	**11**	**0.70**	0.92

Method	**Synthetic Control**			**Toe Segmentation1**			**Trace**		
	k	NMI	RI	k	NMI	RI	k	NMI	RI
Ground truth	6	-	-	2	-	-	4	-	-
X-means	17	**0.82**	**0.94**	6	0.01	0.50	13	0.37	0.75
G-means	18	0.77	0.90	6	0.00	0.50	6	0.45	0.75
PG-means	2	0.00	0.17	**1**	0.00	0.50	1	0.00	0.25
Dip-means	3	0.76	0.83	**1**	0.00	0.50	2	0.67	0.75
Warm-up + X-means	20	0.61	0.87	6	0.01	0.50	13	0.41	0.77
Warm-up + G-means	17	0.61	0.86	6	0.01	0.50	10	0.41	0.76
Warm-up + PG-means	15	0.67	0.87	**1**	0.00	0.50	5	0.28	0.58
Warm-up + Dip-means	14	0.63	0.86	**1**	0.00	0.50	7	0.45	0.76
DipContrast (ours)	**4**	0.69	0.84	6	**0.04**	**0.52**	**4**	**0.73**	**0.87**

Method	**WordSynonyms**		
	k	NMI	RI
Ground truth	25	-	-
X-means	32	0.45	0.90
G-means	**23**	0.42	0.89
PG-means	1	0.00	0.09
Dip-means	1	0.00	0.09
Warm-up + X-means	63	0.49	**0.91**
Warm-up + G-means	28	0.44	0.90
Warm-up + PG-means	1	0.00	0.09
Warm-up + Dip-means	1	0.00	0.09
DipContrast (ours)	38	**0.52**	**0.91**

Table 5.2.: Clustering performance on UCR datasets.

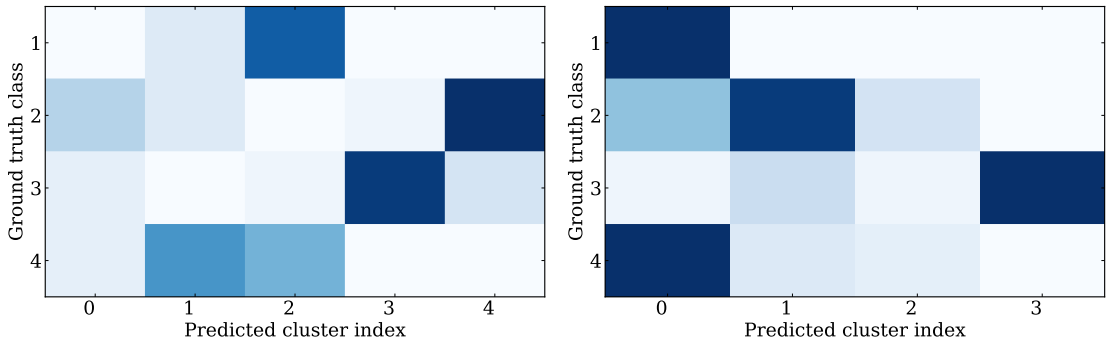
As observed in Table 5.2, selected non-parametric baseline algorithms struggle to provide reasonable clustering for time-series data. Methods such as PG-means and Dip-means frequently produce trivial solution, estimating $k = 1$ and yielding near-zero NMI and RI scores. Conversely, X-means and G-means often overestimate k without capturing the true underlying semantic structures. Applying these baselines to the latent representations obtained after DipContrast's warm-up phase provides a noticeable performance gain across most datasets. This demonstrates that the dual-domain contrastive representation alone establishes a more clustering-friendly latent space. For instance, on the CBF dataset,

classical PG-means and Dip-means both collapse to one cluster with an NMI of 0.00. However, when applied to our warm-up representations, their NMI scores jump to 0.42 and 0.65, respectively. Despite this empirical advantage, the baseline algorithms remain constrained in their ability to reasonably estimate the underlying cluster count.

DipContrast outperforms the baselines, providing reasonable estimates for k while simultaneously achieving better clustering quality. On 29 of the 40 UCR datasets, DipContrast yielded the highest NMI, denoted in bold as the best score among all evaluated algorithms. Examining the results for these 29 datasets reveal three different profiles:

i) High Alignment (Accurate k, High NMI): On datasets such as CBF, GunPointOldVersusYoung, Plane, and Trace, DipContrast captures the ground-truth number of clusters (k) perfectly or near-perfectly while achieving NMI scores ≥ 0.70. For CBF dataset it even achieved perfect clustering with an NMI of 1.00. This represents the ideal scenario where the latent space directly and accurately mirrors the underlying semantic classes.

ii) Meaningful Divergence (Deviated k, Competitive/High NMI): Across a number of the datasets evaluated, the estimated k by DipContrast diverges from the established ground truth, while maintaining high NMI scores relative to the baselines. To interpret this outcome, we align with the rationale established in DipDECK: while we find a divergent number of clusters, the ones we find, often better represent the underlying ground truth.



(a) DipContrast result (b) G-means result

Figure 5.1.: Comparison of confusion matrices for the FaceFour dataset.

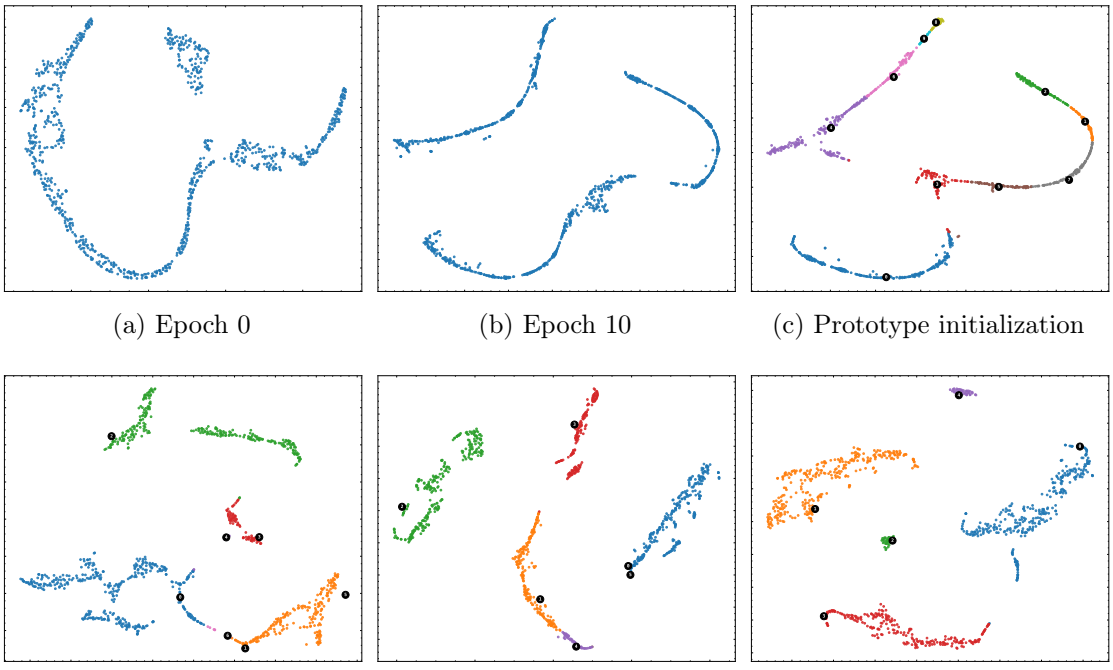
When k is overestimated, DipContrast partitions a ground-truth cluster into multiple partitions or subclusters. For example, predicting $k = 31$ instead of the ground truth of 18 on the Fungi dataset still yields a high NMI of 0.87. This behavior is also evident in the FaceFour dataset, where DipContrast achieves an NMI of 0.54 despite overestimating $k = 5$ (Figure 5.1a). In these matrices, the rows stand for the ground-truth classes (labeled 1 to 4), while the columns show the predicted cluster indices (labeled 0 to 4), with the darkness of a square representing the number of data points. Looking closely at

Figure 5.1a, row 4 of the confusion matrix shows that a single ground-truth cluster is cleanly split into two predicted clusters (columns 1 and 2). Furthermore, columns 3 and 4 map almost exclusively to singular ground-truth classes. There are, of course, various data points assigned to the wrong clusters, but a comparison to the confusion matrix of G-means (Figure 5.1b) highlights the advantage of DipContrast. While G-means correctly estimates $k = 4$, column 0 incorrectly merges three distinct ground-truth classes, resulting in a lower NMI of 0.45.

Conversely, when k is underestimated, DipContrast merges different ground-truth classes that share highly similar temporal patterns. This is demonstrated on the ProximalPhalanxTW dataset, where it predicts $k = 3$ (true $k = 6$) but maintains an NMI of 0.69, and on the Fish dataset, yielding an NMI of 0.43 despite predicting $k = 5$ (true $k = 7$).

iii) Complex Feature Spaces (High relative NMI, but poor absolute): On select datasets (e.g., ECG200, ShapeletSim), DipContrast yields a low absolute NMI despite outperforming baselines. This suggests that contrastive augmentations may perturb class-defining temporal features, causing improper alignment and poor separation.

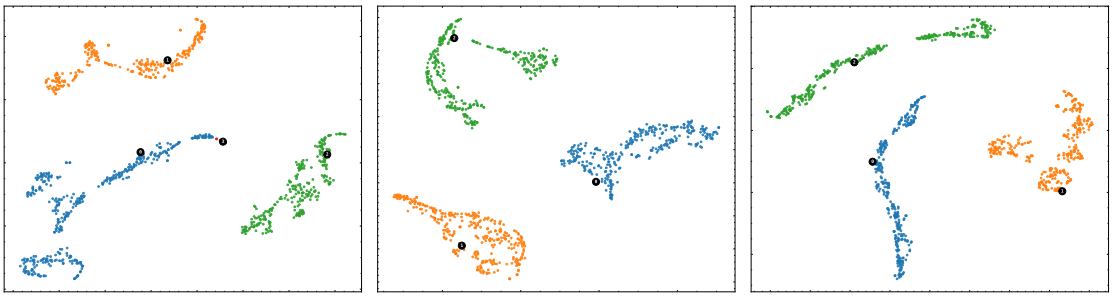
Latent Space Evolution and Convergence Visualizing the training progression on the CBF dataset (ground truth $k = 3$) illustrates how the latent space organizes over time.



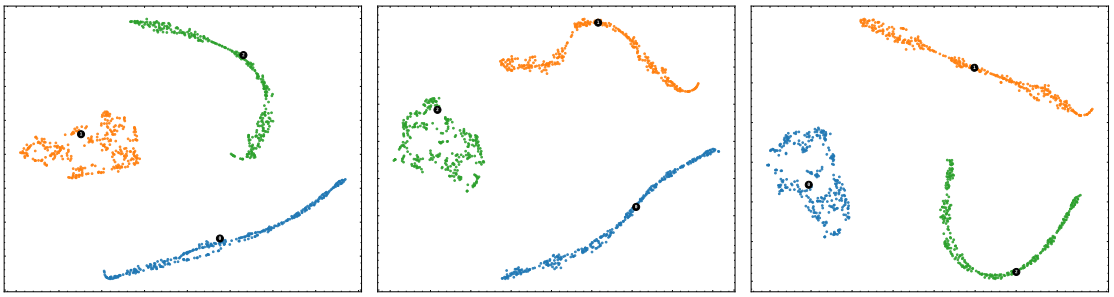
(a) Epoch 0 (b) Epoch 10 (c) Prototype initialization

(d) Epoch 30 (e) Epoch 35 (f) Epoch 40

Figure 5.2.: t-SNE visualizations of CBF dataset training steps (Part 1). (a–b) Warm-up stage; (c) Prototype initialization; (d–f) Training stage.



(a) Epoch 50 (b) Epoch 100 (c) Epoch 150



(d) Epoch 200 (e) Epoch 250 (f) After convergence

Figure 5.3.: t-SNE visualizations of CBF dataset training steps (Part 2). (a–e) Training stage; (f) Latent space after convergence.

As illustrated by the t-SNE [58] visualizations, during the initial warm-up phase (Figure 5.2a–b), the embeddings remain widely dispersed, with no clearly defined decision boundaries. The instance-level contrastive loss establishes a friendly latent space to facilitate stable prototype initialization. Following prototype initialization (Figure 5.2c), the model is configured with an overestimated number of clusters ($k_{\text{init}} \gg k$), and the latent space is initially partitioned into micro-clusters.

From epoch 30 to 50 (Figure 5.2d–f and Figure 5.3a), this latent manifold is progressively refined during the *E-steps*. The non-parametric clustering mechanism evaluates unimodality by projecting candidate clusters onto a single line and applying the Dip-test, steadily merging those that exhibit unimodal distributions.

Throughout the subsequent training stages, the *M-steps* utilize the DipProtoLoss to encourage latent embeddings to align closely with their assigned prototypes. As evident in (Figure 5.3b–e), the latent space stabilizes with well-separated clusters, followed by a progressive refinement of intra-class concentration around the learned prototypes. The final embedding (Figure 5.3f) confirms convergence to a stable configuration with well-isolated clusters, consistent with the ground-truth class cardinality $k = 3$, suggesting that the model has learned a discriminative and semantically coherent latent space.

5.4.2. Smart meter dataset

Method	Smart Meter Dataset		
	k	Silhouette Score	Davies-Bouldin Score
X-means	100	0.03	2.77
G-means	100	0.03	2.77
PG-means	*	*	*
Dip-means	*	*	*
Warm-up + X-means	100	0.32	0.87
Warm-up + G-means	100	0.32	0.88
Warm-up + PG-means	*	*	*
Warm-up + Dip-means	*	*	*
DipContrast (ours)	14	**0.88**	**0.33**

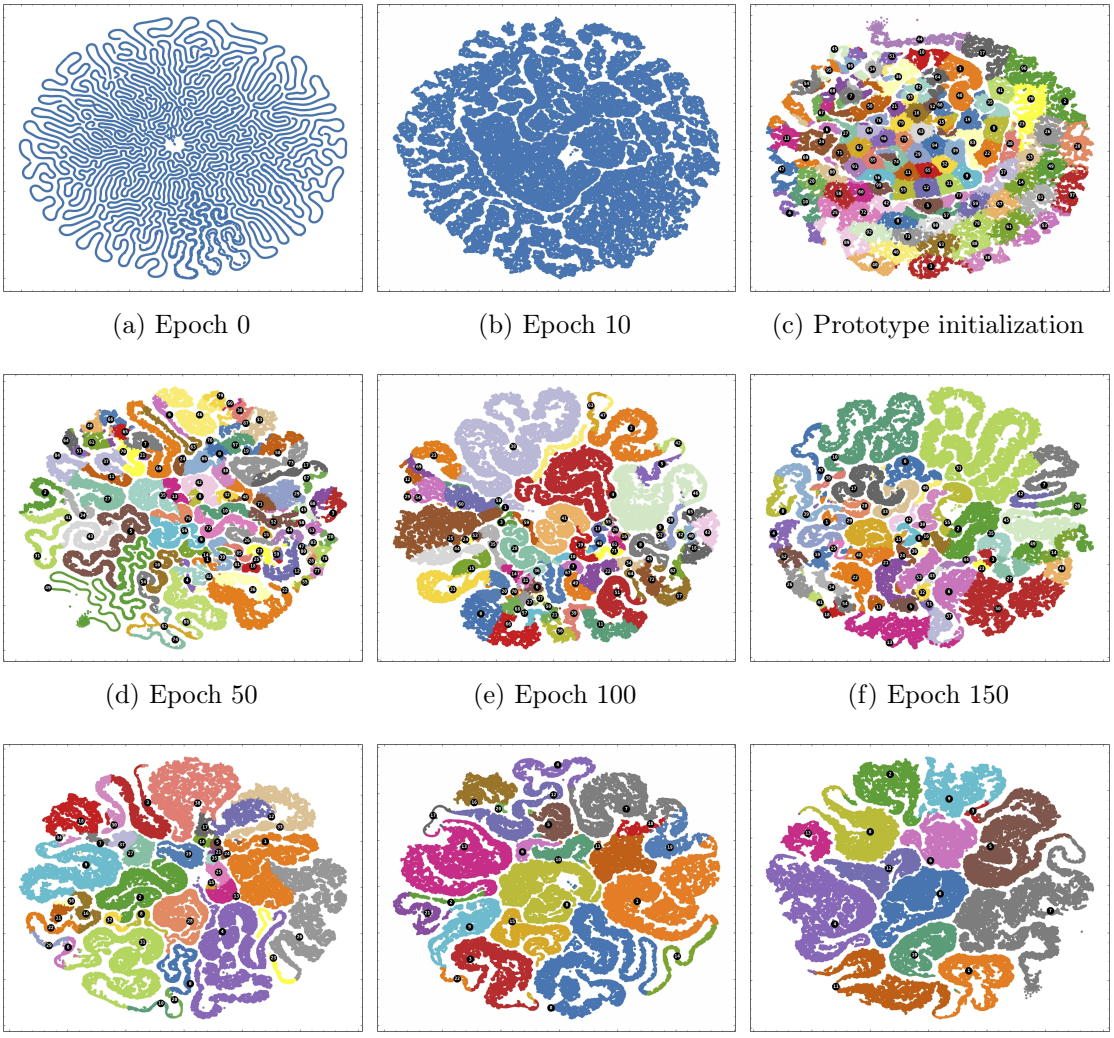
Note: An asterisk (∗) indicates that the algorithm either encountered a runtime error or was aborted after 24 hours of execution.

Table 5.3.: Clustering performance on smart meter dataset.

The results obtained for the smart meter dataset are summarized in Table 5.3. It should be noted that evaluations for the first four methods (X-means, G-means, PG-means, and Dip-means) were conducted on the original data points, whereas evaluations for the last five methods (those utilizing a warm-up phase and our proposed DipContrast) were performed on the embedded data points. Notably, DipContrast outperforms all baselines with a Silhouette score of 0.88, a DB score of 0.33, and a refined cluster count of $k = 14$. In contrast, baseline algorithms struggle with the scale of the data, as both PG-means and Dip-means fail to produce any clustering results on either the raw data or the learned latent representations after warm-up stage. While X-means and G-means successfully execute, they remain fixed at the initial overestimation of $k = 100$ and yield poor scores for the internal metrics. The results further highlight the importance of the initial warm-up phase. Applying X-means and G-means to the latent representations obtained after this stage leads to a substantial improvement, with Silhouette scores increasing from 0.03 to 0.32 and DB scores decreasing from 2.77 to approximately 0.87.

Latent Space Evolution and Convergence The training progression on the smart meter dataset is illustrated in Figure 5.4. As evident in the visualizations, during the initial warm-up phase (Figure 5.4a–b), the embeddings form a continuous and non-separable manifold with no clear cluster structure. Following the warm-up stage, the model is configured with an overestimated number of clusters (Figure 5.4c), and prototypes are initialized. DipContrast's non-parametric clustering mechanism steadily merges valid candidate clusters, ensuring the over-partitioned latent feature space is progressively refined into cohesive groupings (Figure 5.4d–f). As training advances through the latter stages (Figure 5.4g–h), this cluster merging process reorganizes the latent space into a semantically coherent configuration. Consequently, the latent space stabilizes and is iteratively refined into well-separated and compact clusters parameterized by the

dispersion and concentration of individual prototypes. By the conclusion of training, the latent representations converge into 14 distinct clusters (Figure 5.4i).



(a) Epoch 0 (b) Epoch 10 (c) Prototype initialization

(d) Epoch 50 (e) Epoch 100 (f) Epoch 150

(g) Epoch 200 (h) Epoch 250 (i) After convergence

Figure 5.4.: t-SNE visualizations of smart meter dataset training steps. (a–b) Warm-up stage; (c) Prototype initialization; (d–h) Training stage; (i) Latent space after convergence.

5.5. Qualitative Analysis

In this section, we move beyond standard performance benchmarks to evaluate the internal mechanics and operational characteristics of DipContrast. We provide an ablation study to validate the necessity of our proposed components, followed by a parameter sensitivity

analysis to evaluate the robustness of the clustering mechanism against variations in key hyperparameters.

5.5.1. Ablation Study

Through an ablation study on the 40 selected UCR datasets, we evaluated the contribution and effectiveness of each core component of DipContrast. We compare the complete DipContrast framework against four ablation variants:

- **w/o warm-up**: Omits the initial instance-level contrastive warm-up phase, meaning prototypes are initialized on a less stable and cluster-friendly latent space.
- **w/o** $\mathcal{L}^{\text{DipProto}}$: Excludes the Dip-Guided Prototypical Loss from $\mathcal{L}_t^{\text{training}}$ and $\mathcal{L}_f^{\text{training}}$, thereby relying solely on the instance-level contrastive loss during training stage.
- **w/o** $\mathcal{L}_{tf}^{\text{Ins}} + \mathcal{L}_{tf}^{\text{training}}$: Removes the cross-domain consensus components from both the initial warm-up and the subsequent training stage, preventing the alignment between the time and frequency representations of the same sample.
- **w/o** $\mathcal{L}_f^{\text{Ins}} + \mathcal{L}_f^{\text{training}}$: Completely removes the frequency domain and its accompanying optimization objectives from both the initial warm-up and the subsequent training stage, resulting in a model that operates solely within the time domain.

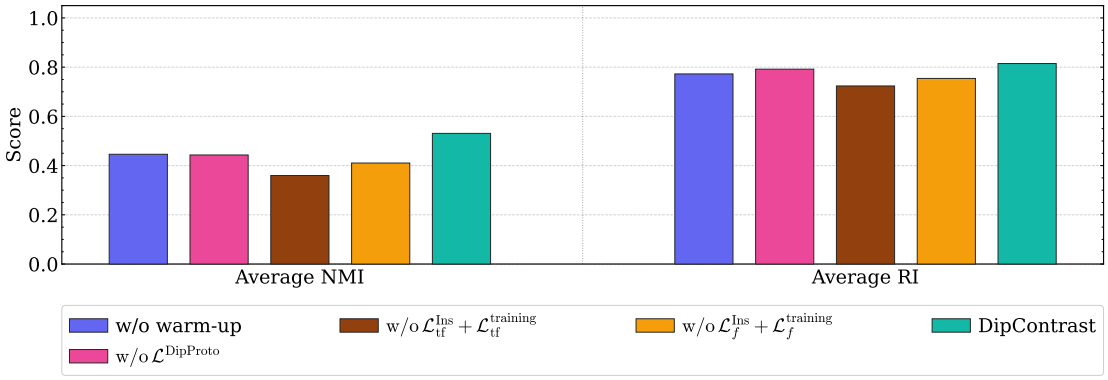


Figure 5.5.: Average NMI and RI of the selected UCR datasets for ablations of DipContrast.

As illustrated in Figure 5.5, the complete DipContrast framework achieves the highest average NMI and average RI scores across the evaluated datasets. The most severe degradation in clustering performance occurs when the cross-domain consensus components ($\mathcal{L}_{tf}^{\text{Ins}} + \mathcal{L}_{tf}^{\text{training}}$) are excluded. This performance drop indicates that cross-domain agreement is essential for learning robust latent representations.

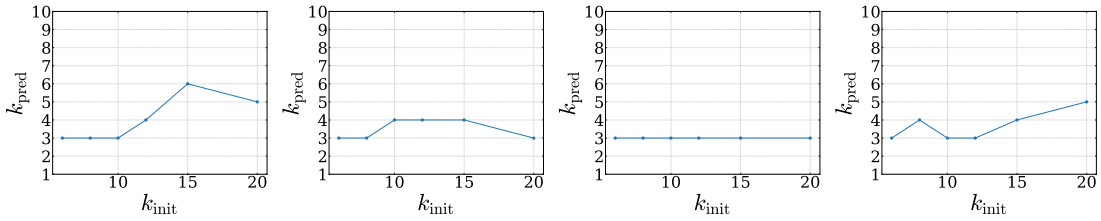
Furthermore, completely removing the frequency domain ($\mathcal{L}_f^{\text{Ins}}+\mathcal{L}_f^{\text{training}}$) yields a similar decline in performance, and underscores the necessity of the dual-domain architecture. It clearly demonstrates that the frequency view contributes essential complementary features missed by a purely time-domain encoder.

Finally, omitting either the warm-up phase or the $\mathcal{L}^{\text{DipProto}}$ objective results in noticeable, but slightly smaller, decrease in both NMI and RI. This confirms that establishing a friendly latent space prior to prototype initialization is a necessary step for achieving optimal clustering results.

5.5.2. Parameter Sensitivity and Robustness

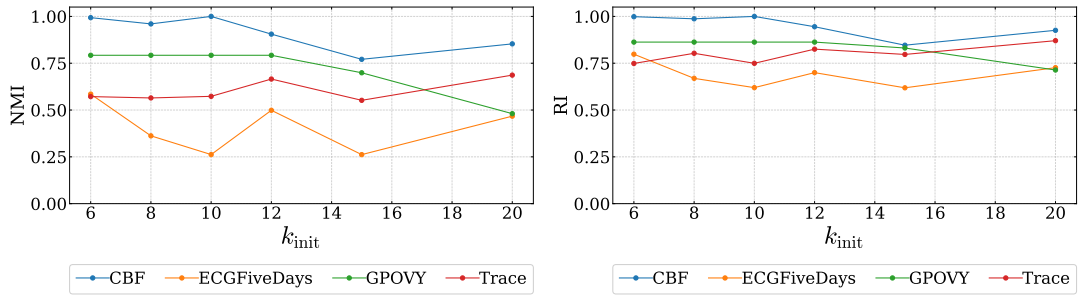
To evaluate the robustness of DipContrast, we analyze the impact of its primary hyperparameters on clustering performance. We focus on two core parameters: the initial overestimation of the number of clusters (k_{init}), and the loss balancing coefficient (λ).

Impact of k_{init} selection: We analyze the impact of k_{init} across four datasets: CBF, ECGFiveDays, GunPointOldVersusYoung (GPOVY), and Trace. The selected datasets contain varying numbers of ground-truth clusters: 3 for CBF, 2 for ECGFiveDays and GPOVY, and 4 for Trace. For this analysis, all other hyperparameters were kept constant, and only k_{init} was varied within the range of $[6, 20]$.



(a) CBF (b) ECGFiveDays (c) GPOVY (d) Trace

Figure 5.6.: $k_{predicted}$ with increasing k_{init}.

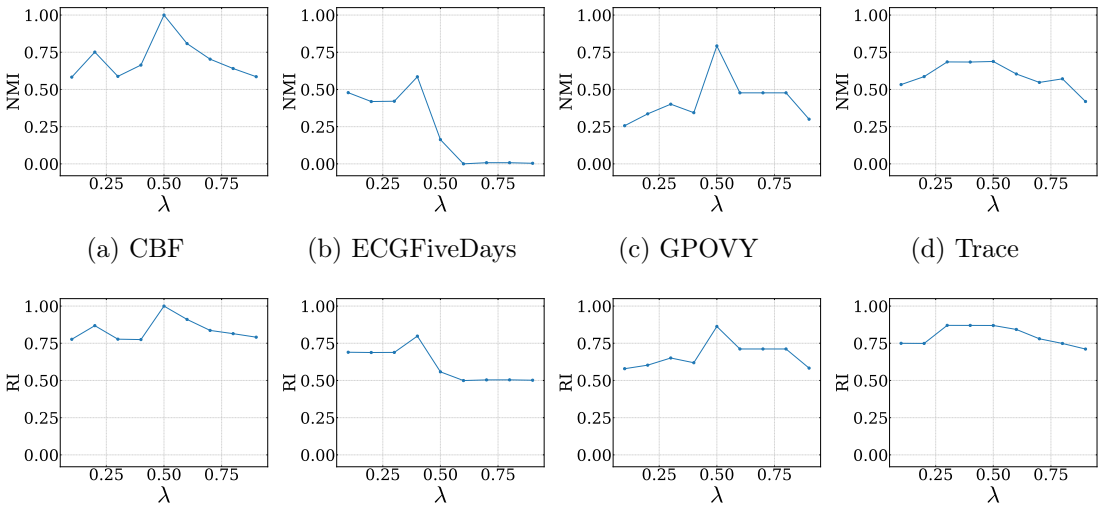


(a) NMI with increasing k_{init} (b) RI with increasing k_{init}

Figure 5.7.: NMI and RI with increasing k_{init}.

As illustrated in Figure 5.6 and Figure 5.7, DipContrast demonstrates empirical robustness to k_{init} overestimation. For the majority of the evaluated datasets (specifically GPOVY, ECGFiveDays, and Trace), the predicted cluster count shown in Figure 5.6 deviates by at most one or two clusters from the ground truth even at the higher end of the evaluated range. Assessing the progression of the NMI and RI metrics in Figure 5.7 indicates that the overall clustering quality remains robust and near-optimal, despite the deviation. The CBF dataset presents the only notable exception, where the framework perfectly estimates the true cluster count ($k = 3$) for $k_{\text{init}} \leq 10$, but exhibits overestimation (predicting 4–6 clusters) at increasingly large initializations.

Impact of λ selection: The loss balancing coefficient λ dictates the trade-off between the instance-level contrastive term and the prototypical loss term. Larger λ emphasizes the instance-level contrastive term relative to the prototypical term. We evaluated the DipContrast's sensitivity to this parameter across the CBF, ECGFiveDays, GPOVY, and Trace datasets.



(a) CBF (b) ECGFiveDays (c) GPOVY (d) Trace

(e) CBF (f) ECGFiveDays (g) GPOVY (h) Trace

Figure 5.8.: NMI (top) and RI (bottom) as a function of the loss balancing coefficient λ.

As illustrated in Figure 5.8, the impact of the balancing coefficient λ reveals a characteristic inverted-U trend across the evaluated metrics. While different datasets exhibit varying sensitivities to λ, optimal clustering results are generally observed around 0.5. When λ is set too low, the model underutilizes the instance-level contrastive loss, which is essential for learning discriminative latent features. Conversely, setting λ excessively high over-emphasizes instance discrimination at the expense of the DipProtoLoss. In this scenario, aggressive instance-wise repulsion pushes the embeddings apart, preventing them from cohesively grouping around their respective prototypes and yielding sub-optimal cluster assignments. This demonstrates that a carefully balanced λ is critical to simul-

taneously preserving individual sample variance and grouping them into semantically meaningful and well-separated clusters.

5.6. Shortcomings

While DipContrast demonstrates strong empirical performance across various datasets, there exist some shortcomings.

In deep learning literature, random seeds are utilized primarily to make computational results reproducible. However, the choice of a random seed can affect results in non-trivial ways. Thus, treating random seeds as optimizable hyperparameters remains an ongoing debate. Despite its counterintuitive nature, this practice is becoming increasingly common. Recent literature highlights that even with identical hyperparameter values, distinct random seeds can lead to substantially different outcomes [59]. Specifically, the choice of seed influences factors such as weight initialization, which contributes to the variance in a model's final performance.

This phenomenon is highly relevant to our proposed framework as the seed dictates how the initial model weights are generated, and our time-series augmentation strategy (e.g., jittering, scaling, and frequency component masking) is inherently stochastic. Consequently, we reported our evaluations using a fixed seed, which ensures reproducibility while presenting the best clustering performance achieved by the model. Highlighting the contrast between fixed and random initializations, we conducted 20 unseeded trials across 10 datasets to serve as a comparison against the optimal seeded performance. Based on the results detailed in Table 5.4, two profiles emerge:

- **High Variance:** On the CBF dataset, a seeded run perfectly predicts the true cluster count of 3 with an NMI of 1.00. However, the unseeded evaluation averages a predicted count of 4.55 ± 0.69 with a significantly lower NMI of 0.78 ± 0.10. A similar degradation occurs with the Trace dataset, where the seeded run achieves an NMI of 0.73, while the unseeded runs yield a fluctuating cluster count of 4.35 ± 0.81 and an NMI drop to 0.65 ± 0.07. For the Fungi dataset, the predicted cluster count varies widely in unseeded runs (30.75 ± 5.16) compared to the fixed prediction of 31.
- **High Stability:** Conversely, datasets such as ShapesAll and PigAirwayPressure remain remarkably stable regardless of the seed. PigAirwayPressure maintains an exact NMI of 0.77 ± 0.00 across runs, and ShapesAll consistently predicts exactly 144 clusters with an NMI of 0.74 ± 0.01.

Dataset	K_{true}	$k_{\text{predicted}}$		NMI		RI	
		w/ seed	w/o seed	w/ seed	w/o seed	w/ seed	w/o seed
CBF	3	3	4.55±0.69	1.00	0.78±0.10	1.00	0.87±0.06
DiatomSizeReduction	4	3	2.30±0.47	0.81	0.64±0.11	0.92	0.76±0.10

Continued on next page

Table 5.4 – continued from previous page

Fungi	18	31	30.75 ± 5.16	0.87	0.83±0.02	0.97	0.96±0.01
GunPointOldVersusYoung	2	3	2.75±0.44	0.79	0.65±0.19	0.86	0.79±0.11
PigAirwayPressure	52	165	164.30 ± 2.92	0.77	0.77±0.00	0.98	0.98±0.00
PigArtPressure	52	147	153.95 ± 4.87	0.83	0.84±0.01	0.98	0.99±0.00
PigCVP	52	163	150.20 ± 5.33	0.78	0.77±0.01	0.98	0.98±0.00
Plane	7	7	8.35±0.93	0.93	0.91±0.02	0.98	0.97±0.01
ShapesAll	60	144	144.00 ± 0.00	0.74	0.74±0.01	0.98	0.98±0.00
Trace	4	4	4.35±0.81	0.73	0.65±0.07	0.87	0.83±0.04

Table 5.4.: Comparison of predicted k, NMI, and RI for DipContrast evaluation with seed vs. without seed. W/o seed: mean ± standard deviation over 20 runs.

Another limitation, as demonstrated in the robustness experiments, is the framework's sensitivity to the initial cluster count, k_{init}. For certain datasets, when initialized with a very large k_{init}, DipContrast tends to overestimate the final number of clusters ($k_{\text{predicted}}$). This can be attributed to how the loss functions are defined. The "push-pull" mechanism describes the opposing and complementary forces exerted on the latent embeddings during training. These forces are generated by the interplay between the instance-level contrastive loss and the DipProtoLoss. At a granular level, the model uses instance loss to shape the local geometry of the embedding space. Specifically, this objective "pulls" the representation of an anchor z_i closer to its specific augmented view $\tilde{z}_i$. Concurrently, the DipProtoLoss exerts forces to create clusters and pulls every instance z_i towards its assigned cluster centroid c_k. While this happens, defining a very large k_{init} seems to have an impact on $k_{\text{predicted}}$.

A further limitation arises from how the framework updates cluster centers at every epoch. During the *E-step* of the optimization cycle, DipContrast recomputes the prototypes by running k-means from scratch on the current latent representations. While this ensures that the prototypes accurately reflect the most recent state of the evolving latent space, initializing and hard-assigning prototypes at the beginning of every epoch causes abrupt shifts in the cluster centroids even with k-means++ initialization. The continuous resetting of centroids could interfere with the smooth evaluation of the Dip-test. To mitigate this instability, the architecture could be adapted to utilize a momentum-based mechanism, similar to MoCo [45]. Instead of re-running k-means from scratch, the framework could maintain a slowly updating exponential moving average of the prototypes.

6. Conclusion

In this paper, we proposed **DipContrast**, an end-to-end contrastive learning framework for non-parametric deep time-series clustering. Our framework unifies representation learning and clustering to discover semantic partitions in large-scale, unlabeled data without prior knowledge of the cluster count k. The dual-domain architecture of DipContrast facilitates the extraction of diverse feature sets from both time and frequency views, ensuring a more holistic latent representation. The framework progressively organizes the latent space through a prototype-driven mechanism, dynamically merging candidate clusters based on the evaluation of unimodality via Hartigan's Dip-test.

Empirical evaluations show that DipContrast consistently surpasses the performance of classical non-parametric baselines. In 29 of the 40 selected UCR datasets, our framework outperformed all evaluated algorithms in terms of NMI. DipContrast successfully captured underlying semantic structures, avoiding the trivial solutions or excessive overestimations common to the non-parametric baselines. On the large-scale smart meter dataset, DipContrast achieved a Silhouette score of 0.88, indicating highly cohesive and well-separated clusters. In contrast, baseline algorithms yielded poor internal metric scores or failed to execute successfully on the smart meter data. Ablation studies further confirmed that cross-domain consensus and the integration of the frequency view are essential for learning robust latent representations.

While the current framework demonstrates strong empirical performance, the iterative re-initialization of cluster centroids during the *E-step* can introduce slight instability. Future work will explore momentum-based prototype updates, similar to MoCo, to ensure a smoother non-parametric evolution. In summary, DipContrast provides a robust foundation for DTSC, providing an effective solution for discovering latent patterns and extracting data-driven insights from time-series data.

Bibliography

[1] I. Assent, "Clustering high dimensional data," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 340–350, 2012. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1062

[2] B. Saha, D. Krotov, M. J. Zaki, and P. Ram, "End-to-end differentiable clustering with associative memories," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.

[3] J. Guo, X. Yuan, P. Xu, H. Bai, and B. Liu, "Improved image clustering with deep semantic embedding," *Pattern Recognition Letters*, vol. 130, pp. 225–233, 2020, image/Video Understanding and Analysis (IUVA). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786551830850X

[4] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," *Knowledge-Based Systems*, vol. 197, p. 105841, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705120302112

[5] M. R. Karim, O. Beyan, A. Zappa, I. G. Costa, D. Rebholz-Schuhmann, M. Cochez, and S. Decker, "Deep learning-based clustering approaches for bioinformatics," *Briefings in Bioinformatics*, vol. 22, no. 1, pp. 393–415, 02 2020. [Online]. Available: https://doi.org/10.1093/bib/bbz170

[6] X. Zhu, Y. Zhu, and W. Zheng, "Spectral rotation for deep one-step clustering," *Pattern Recognition*, vol. 105, p. 107175, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320319304753

[7] K. Li, T. Ni, J. Xue, and Y. Jiang, "Deep soft clustering: simultaneous deep embedding and soft-partition clustering," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 5, pp. 5581–5593, May 2023. [Online]. Available: https://doi.org/10.1007/s12652-021-02997-1

[8] L. Wu, L. Yuan, G. Zhao, H. Lin, and S. Z. Li, "Deep clustering and visualization for end-to-end high-dimensional data analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 8543–8554, 2023.

[9] W. Guo, W. Zhang, Z. Zhang, P. Tang, and S. Gao, "Deep temporal iterative clustering for satellite image time series land cover analysis," *Remote Sensing*, vol. 14, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/15/3635

[10] J. Kim and N. Moon, "A deep bidirectional similarity learning model using dimensional reduction for multivariate time series clustering," *Multimedia Tools and Applications*, vol. 80, pp. 34 269 – 34 281, 2021.

[11] C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, "Dip-based deep embedded clustering with k-estimation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 903–913. [Online]. Available: https://doi.org/10.1145/3447548.3467316

[12] Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, P. S. Yu, and L. He, "Deep clustering: A comprehensive survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 4, pp. 5858–5878, 2025.

[13] X. Wei, Z. Zhang, H. Huang, and Y. Zhou, "An overview on deep clustering," *Neurocomputing*, vol. 590, p. 127761, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224005320

[14] X. Li, Z. Chen, L. K. M. Poon, and N. L. Zhang, "Learning latent superstructures in variational autoencoders for deep multidimensional clustering," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=SJgNwi09Km

[15] F. Ye and A. Bors, "Deep mixture generative autoencoders," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 5789 – 5803, 10 2022.

[16] S. Zhou, H. Xu, Z. Zheng, J. Chen, Z. Li, J. Bu, J. Wu, X. Wang, W. Zhu, and M. Ester, "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," *ACM Comput. Surv.*, vol. 57, no. 3, Nov. 2024. [Online]. Available: https://doi.org/10.1145/3689036

[17] P. Trirat, Y. Shin, J. Kang, Y. Nam, J. Na, M. Bae, J. Kim, B. Kim, and J. Lee, "Universal time-series representation learning: A survey," *CoRR*, vol. abs/2401.03717, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.03717

[18] E. Eldele, M. Ragab, Z. Chen, M. Wu, C. K. Kwoh, X. Li, and C. Guan, "Time-series representation learning via temporal and contextual contrasting," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z.-H. Zhou, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2021, pp. 2352–2359, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2021/324

[19] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, and B. Xu, "Ts2vec: Towards universal representation of time series," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8980–8987, Jun. 2022. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20881

[20] J. Liu and S. Chen, "Timesurl: Self-supervised contrastive learning for universal time series representation learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 12, pp. 13 918–13 926, Mar. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29299

[21] D. Luo, W. Cheng, Y. Wang, D. Xu, J. Ni, W. Yu, X. Zhang, Y. Liu, Y. Chen, H. Chen, and X. Zhang, "Time series contrastive learning with information-aware augmentations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, pp. 4534–4542, Jun. 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/25575

[22] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/53c6de78244e9f528eb3e1cda69699bb-Paper.pdf

[23] L. Yang and S. Hong, "Unsupervised time-series representation learning with iterative bilinear temporal-spectral fusion," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 25 038–25 054. [Online]. Available: https://proceedings.mlr.press/v162/yang22e.html

[24] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 478–487. [Online]. Available: https://proceedings.mlr.press/v48/xieb16.html

[25] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018.

[26] A. Alqahtani, M. Ali, X. Xie, and M. W. Jones, "Deep time-series clustering: A review," *Electronics*, vol. 10, no. 23, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/23/3001

[27] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5147–5156.

[28] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part*

XIV. Berlin, Heidelberg: Springer-Verlag, 2018, p. 139–156. [Online]. Available: https://doi.org/10.1007/978-3-030-01264-9_9

[29] N. Madiraju, S. M. Sadat, D. Fisher, and H. Karimabadi, "Deep temporal clustering : Fully unsupervised learning of time-domain features," 02 2018. [Online]. Available: https://www.researchgate.net/publication/322950204_Deep_Temporal_Clustering_Fully_Unsupervised_Learning_of_Time-Domain_Features

[30] Y. Zhong, D. Huang, and C.-D. Wang, "Deep temporal contrastive clustering," *Neural Process. Lett.*, vol. 55, no. 6, p. 7869–7885, May 2023. [Online]. Available: https://doi.org/10.1007/s11063-023-11287-0

[31] S. Lee, C. Choi, and Y. Son, "Deep time-series clustering via latent representation alignment," *Knowledge-Based Systems*, vol. 303, p. 112434, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705124010682

[32] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, p. 727–734.

[33] G. Hamerly and C. Elkan, "Learning the k in k-means," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16. MIT Press, 2003. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2003/file/234833147b97bb6aed53a8f4f1c7a7d8-Paper.pdf

[34] Y. Feng and G. Hamerly, "Pg-means: learning the number of clusters in data," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19. MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2006/file/a9986cb066812f440bc2bb6e3c13696c-Paper.pdf

[35] A. Kalogeratos and A. Likas, "Dip-means: an incremental clustering method for estimating the number of clusters," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/a8240cb8235e9c493a0c30607586166c-Paper.pdf

[36] X. Ye, J. Zhao, L. Zhang, and L. Guo, "A nonparametric deep generative model for multimanifold clustering," *IEEE Transactions on Cybernetics*, vol. 49, no. 7, pp. 2664–2677, 2019.

[37] M. Ronen, S. E. Finder, and O. Freifeld, "Deepdpm: Deep clustering with an unknown number of clusters," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 9861–9870.

[38] D. Mautz, C. Plant, and C. Böhm, "Deep embedded cluster tree," in *2019 IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 1258–1263.

[39] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 539–546 vol. 1.

[40] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[41] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf

[42] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 297–304. [Online]. Available: https://proceedings.mlr.press/v9/gutmann10a.html

[43] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.

[44] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1597–1607. [Online]. Available: https://proceedings.mlr.press/v119/chen20j.html

[45] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9726–9735.

[46] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent a new approach to self-supervised learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[47] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[48] X. Chen and K. He, "Exploring simple siamese representation learning," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 745–15 753.

[49] T. Chen, C. Luo, and L. Li, "Intriguing properties of contrastive losses," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021.

[50] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks," in *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ser. ICMI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 216–220. [Online]. Available: https://doi.org/10.1145/3136755.3136817

[51] F. Peng, J. Luo, X. Lu, S. Wang, and F. Li, "Cross-domain contrastive learning for time series clustering," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, pp. 8921–8929, Mar. 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/28740

[52] J. A. Hartigan and P. M. Hartigan, "The Dip Test of Unimodality," *The Annals of Statistics*, vol. 13, no. 1, pp. 70 – 84, 1985. [Online]. Available: https://doi.org/10.1214/aos/1176346577

[53] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, "Prototypical contrastive learning of unsupervised representations," in *International Conference on Learning Representations (ICLR)*, 2021. [Online]. Available: https://iclr.cc/virtual/2021/poster/3090

[54] C. Leiber, L. G. M. Bauer, B. Schelling, C. Böhm, and C. Plant, "Dip-based deep embedded clustering with k-estimation," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 903–913. [Online]. Available: https://doi.org/10.1145/3447548.3467316

[55] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

[56] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, "Time-series clustering – a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437915000733

[57] C. Leiber, L. Miklautz, C. Plant, and C. Böhm, "Benchmarking deep clustering algorithms with clustpy," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2023, pp. 625–632.

[58] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[59] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," 2020.

Acronyms

AE Autoencoder. 5

BIC Bayesian Information Criterion. 8

BYOL Bootstrap Your Own Latent. 14

CDCC Cross-Domain Contrastive Learning. 23–25, 33

CNN Convolutional Neural Network. 19, 20, 24

DB Davies-Bouldin. 39, 48

DCAE Deep Convolutional Autoencoder. 9

DDC Deep Density-based Clustering. 9

DeepECT Deep Embedded Cluster Tree. 9

DipContrast Contrastive Learning Framework for Non-Parametric Deep Time-Series Clustering. iii, v, 2, 3, 9, 16, 23–33, 35, 39, 40, 44–46, 48–55, 67, 68

DipDECK Dip-based Deep Embedded Clustering with k-Estimation. 9, 23, 29, 30, 45

DipProtoLoss Dip-Guided Prototypical Loss. 26, 28, 31–33, 47, 50, 52, 54

DNN Deep Neural Network. 5, 18

DTC Deep Temporal Clustering. 7

DTCC Deep Temporal Contrastive Clustering. 7

DTIC Deep Temporal Iterative Clustering. 7

DTSC Deep Time-Series Clustering. 1, 6–9, 55

EM Expectation-Maximization. 25

GAN Generative Adversarial Network. 5, 6

GMM Gaussian Mixture Models. 6

GNN Graph Neural Network. 5

IME Intelligentes Messgerät in der erweiterten Konfiguration. 36

IMS Intelligentes Messgerät in der Standardkonfiguration. 36

InfoNCE Information Noise-Contrastive Estimation. 13, 15

LSTM Long Short-Term Memory. 18–20, 23

LTVAE Latent Tree Variational Autoencoder. 5

MLP Multilayer Perceptron. 14, 24

MoCo Momentum Contrast. 14, 54, 55

MVAE Mixtures of Variational Autoencoders. 5

NCE Noise-Contrastive Estimation. 13

NMI Normalized Mutual Information. 37, 44–46, 50–53, 55

NT-Xent Normalized Temperature-Scaled Cross-Entropy. 15

PCL Cross-Domain Contrastive Learning. 23, 25, 26, 31, 32

ReLU Rectified Linear Unit. 14, 21, 24

RI Rand Index. 37, 38, 44, 50–52

RNN Recurrent Neural Network. 19

SimCLR Simple Contrastive Learning of Representations. 14, 15, 40

SimSiam Simple Siamese. 14

SSL Self-Supervised Learning. 13

SwAV Swapped Augmentations and Views. 14

VAE Variational Autoencoder. 5, 6, 8

A. Appendix

Parameter Category	**Search Space**
Optimization & Contrastive Learning	
Learning Rate	$\{10^{-5}, 10^{-3}, 10^{-2}\}$
Weight Decay	$\{10^{-9}, 10^{-6}, 10^{-4}\}$
Adam β_1	$\{0.9, 0.95\}$
Adam β_2	$\{0.99, 0.999\}$
Instance Temperature (τ_{ins})	$\{0.2, 0.5, 0.6\}$
Data Augmentation	
Jitter σ (σ_{jitter})	$\{0.5, 0.8, 1.0\}$
Scaling σ (σ_{scaling})	$\{0.2, 0.5, 0.8\}$
Permutation Window Size (W)	$\{5, 8, 16\}$
Add Perturbation Ratio	$\{0.05, 0.1, 0.15\}$
Remove Perturbation Ratio	$\{0.05, 0.1, 0.15\}$
Time Encoder	
Output Size	$\{256, 512, 1024\}$
Hidden Size	$\{512, 1024, 2048\}$
Number of Layers	$\{2, 3, 4\}$
Dropout Probability	$\{0.3, 0.5, 0.7\}$
Frequency Encoder	
Kernel Size	$\{4, 8, 16\}$
Stride	$\{1, 2\}$
Output Channels	$\{32, 64, 128\}$
Dropout Probability	$\{0.2, 0.3, 0.4\}$
Clustering (DipContrast)	
λ	$\{0.4, 0.45, 0.5, 0.55, 0.6\}$
α	$\{5.0, 10.0, 20.0\}$
γ	$\{80.0, 100.0, 120.0, 200.0\}$

Continued on next page

Table A.1 – continued from previous page

Parameter Category	**Search Space**
Merge Interval (T_{merge})	$\{2, 3, 5\}$
Dip Threshold (τ_{dip})	$\{0.8, 0.9, 0.95\}$
Cluster Balance Factor	$\{1.5, 2.0, 3.0\}$

Table A.1.: Hyperparameter search space utilized for DipContrast.