Representation learning and forecasting for inter-related time series

Jingwei Zuo

université PARIS-SACLAY

Apprentissage de représentations et prédiction pour des séries-temporelles inter-dépendantes

Representation learning and forecasting for inter-related time series

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique
Graduate School: Informatique et sciences du numérique
Référent: Université de Versailles-Saint-Quentin-en-Yvelines (UVSQ)

Thèse préparée dans l'unité de recherche **DAVID** (Université Paris-Saclay, UVSQ), sous la direction de **Karine ZEITOUNI**, Professeure, le co-encadrement de **Yehia TAHER**, Maître de conférences.

Thèse soutenue à Versailles, le 9 mai 2022, par

Jingwei ZUO

Composition du jury

Antoine Cornuéjols Professeur des universités, AgroParisTech	Président
Angela Bonifati Professeure, Université Claude Bernard Lyon 1	Rapportrice & Examinatrice
Engelbert Mephu Nguifo Professeur, Université Clermont Auvergne	Rapporteur & Examinateur
Romain Tavenard Maître de conférences (HDR), Université de Rennes 2	Examinateur
Karine Zeitouni Professeure, UVSQ, Université Paris-Sacaly	Directrice de thèse

THESE DE DOCTORAT

NNT : 2022UPASG038

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage de représentations et prédiction pour des séries-temporelles inter-dépendantes

Mots clés: Séries temporelles, Apprentissage des représentations, Flux de données, Apprentissage semi-supervisé, Séries temporelles géolocalisées, Prévision de trafic

Résumé: Les séries temporelles sont un type de données endémique dans de nombreux domaines d'applications, telles que l'analyse financière, la surveillance de l'environnement ou encore l'astronomie. Du fait de leur structure complexe, les séries temporelles amènent à de nouveaux défis dans le traitement et l'extraction de motifs de ces données. La représentation des séries temporelles joue un rôle déterminant dans les méthodes d'apprentissage et les tâches de fouille de données. Peu de méthodes tiennent compte des interdépendances entre séries temporelles différentes.

Dans cette thèse, nous étudions différentes approches de représentation de séries temporelles capables de s'adapter pour diverses tâches de fouille de séries temporelles et à capturer ces interdépendances. Nous étudions la représentation des séries temporelles pour la classification, l'apprentissage semi-supervisé et la prédiction pour des séries temporelles collectées à partir de diverse applications sous différentes formes, i.e, flux de séries temporelles, séries temporelles multivariées (MTS) et séries temporelles géolocalisées (GTS). Nos principales contributions sont les suivantes: (i) l'apprentissage dynamique de la représentation de séries temporelles dans un contexte de flux; (ii) l'apprentissage semi-supervisé de représentation de séries temporelles multivariées; (iii) l'apprentissage de la représentation de séries temporelles géolocalisées pour la prévision du trafic.

Title: Representation Learning and Forecasting for Inter-related Time Series

Keywords: Time series, Representation learning, Data stream, Semi-supervised learning, Geo-located time series, Traffic forecasting

Abstract: Time series is a common data type that has been applied to enormous real-life applications, such as financial analysis, environmental monitoring, and astronomical discovery. Due to its complex structure, time series raises several challenges in their data processing and mining. The representation of time series plays a key role in data mining tasks and machine learning algorithms for time series. Yet, a few methods consider the interrelation that may exist between different time series.

In this thesis, we will study different time series representation approaches that can be used in various time series mining tasks, while capturing the relationships among them. Particularly, we study the time series representation for classification, semi-supervised learning and forecasting tasks on the time series collected from various application contexts under different forms, i.e., time series stream, multivariate time series (MTS) and geo-located time series (GTS). For these tasks, our main contributions are the following: (i) dynamic time series representation learning in a streaming context; (ii) semi-supervised representation learning in multivariate time series; (iii) geo-located time series representation learning for traffic forecasting.

Acknowledgement

This thesis benefited from the kind support and help from many people.

Before all, I would like to express my sincere and deep gratitude to my advisors, Karine Zeitouni, professor of UVSQ, Universté Paris-Saclay and Yehia Taher, Associated professor of UVSQ, Universté Paris-Saclay, for their guide and help during my three years of thesis work as well as during my master's end-of-study internship. Their patience, kindness and generosity have solved many problems both technically and emotionally. Both of them have been great role models as researchers and mentors. In particular, I admire Karine deeply for her dedication to research and impressive depth/width of knowledge, while Yehia has been of great help for promoting my research work always in simple language and for overcoming many technical issues.

I am very lucky to work in a friendly and pleasant environment within the ADAM research team in DAVID Lab. I would specially thank Dr. Zoubida Kedad for her help during my master's and Ph.D. study. I also want to thank Dr. Stéphane Lopes, Dr. Laurent Yeh, Dr. Nicoleta Preda, Dr. Zaineb Chelly Dagdia, and Dr. Béatrice Finance for their help on both technical issues and constant encouragement during my thesis. I would like to thank all the staff members at the DAVID Lab. I sincerely thank our team assistants, Catherine Le Quere and Chantal Ducoin, for helping me with many administration processes.

Great gratitude is also given to Prof. Philippe Pucheral and Dr. Nicolas Anciaux in Inria Saclay. They led me into the field of Data Science when I decided to continue my master's study in Versailles. I also want to thank Dr. Iulian Sandu Popa for his support during my master's and Ph.D. study.

This dissertation would not have been possible without the support, contribution, and friendship of all my colleagues in DAVID Lab. I am really grateful to meet them for their enthusiasm and their valuable and enjoyable time, in particular: Hafsa EI Hafyani, Mohamad Rihany, Alaa Zreik, Souheir Mehanna, Julien Loudet, Zoé Chevallier, Alexandros Kontarinis, Redouane Bouhamoum, Mohammad Abboud, Li Zhang, Mariem Brahem, Ahmad ktaish, Baudouin Naline, Riham Badra, Lívia Almada Cruz, Hadi Dayekh, Perla Hajjar, Robin Carpentier, Ludovic Javet, Julien Mirval, Riad Ladjel.

I would like to warmly thank jury members in my PhD defense including the two reviewers Prof. Angela Bonifati (Université Claude Bernard Lyon 1), Prof. Engelbert Mephu Nguifo (Université Clermont Auvergne) and Prof. Antoine Cornuéjols (AgroParisTech), Dr. Romain Tavenard (Université de Rennes 2).

I would like to thank L'Institut DATAIA for granting me the scholarship for this Ph.D. and for the support for attending conferences and seminars. I would like to

thank Jean Zay project from IDRIS for providing computational resources for the StreamOps project.

I would also like to thank my friends, in France and China, for their mental support and helpful discussions: these are really important to me.

Last but not least, I am grateful for the continuous support and love of my parents and brother. Their unwavering faith and confidence in me and my abilities have shaped me into the person I am. I will do my best to recompense for their support and comprehension on years of my absence from their side even though they will never be repayable.

Résumé en Français

Les séries temporelles sont un type de données endémique dans de nombreux domaines d'applications, telles que l'analyse financière, le diagnostic médical, la surveillance de l'environnement ou encore l'astronomie. Du fait de leur structure complexe, les séries temporelles amènent à de nouveaux défis dans le traitement et l'extraction de connaissances de ces données. La représentation des séries temporelles joue un rôle déterminant dans les méthodes d'apprentissage et les tâches de fouille de données. Cependant, peu de méthodes tiennent compte des interdépendances entre séries temporelles différentes. De plus, la fouille de séries temporelles nécessite de considérer non seulement les caractéristiques des séries temporelles en termes de complexité des données, mais également les contextes particuliers des applications et la tâche de fouille de données à effectuer. Cela nous permet de construire des représentations spécifiques à la tâche.

Dans cette thèse, nous étudions différentes approches de représentation de séries temporelles capables de s'adapter à diverses tâches de fouille de séries temporelles, tout en capturant les relations entre elles. Nous nous concentrons spécifiquement sur la modélisation des interdépendances entre séries temporelles lors de la construction des représentations, qui peuvent être la dépendance temporelle au sein de chaque source de données ou la dépendance inter-variable entre des sources de données différentes. En conséquence, nous étudions les séries temporelles collectées dans diverses applications sous différentes formes. Tout d'abord, pour tenir compte de la dépendance temporelle entre les observations, nous apprenons la représentation de série temporelle dans un contexte de flux dynamique, où la série temporelle est générée en continu à partir de la source de données. Quant à la dépendance inter-variable, nous étudions les séries temporelles multivariées (MTS) avec des données collectées à partir de plusieurs sources. Enfin, nous étudions le MTS dans le contexte de la ville intelligente, où chaque source de données est associée à une localisation spatiale. Par conséquent, le MTS devient une série temporelle géo-localisée (GTS), pour laquelle la modélisation de la dépendance inter-variable requière la prise en compte de l'information spatiale sous-jacente. De ce fait, pour chaque type de séries temporelles collectées dans des contextes différents, nous proposons une méthode de représentation adaptée aux dépendances temporelles et/ou inter-variables.

Outre la complexité des données provenant des interdépendances des séries temporelles, nous étudions diverses tâches d'apprentissage automatique sur des séries temporelles afin de valider les représentations apprises. Les tâches d'apprentissage étudiées dans cette thèse consistent en la classification de séries temporelles, la prévision de séries temporelles et l'apprentissage semi-supervisé de séries temporelles. Nous montrons comment les représentations apprises sont exploitées dans ces différentes tâches d'apprentissage de séries temporelles et pour des applications distinctes.

Plus précisément, nos principales contributions sont les suivantes. En premier lieu, nous proposons un modèle d'apprentissage dynamique de la représentation des séries temporelles dans le contexte du flux de données, où nous considérons à la fois les caractéristiques des séries temporelles et les défis des flux de données. Nous affirmons et démontrons que le motif de Shapelet, basé sur la forme, est la meilleure représentation dans le contexte dynamique. Par ailleurs, nous proposons un modèle semi-supervisé pour l'apprentissage de représentation dans les séries temporelles multivariées (MTS). Ce modèle considère la dépendance inter-variable dans l'hypothèse réaliste où les annotations de données sont limitées. Enfin, nous proposons un modèle d'apprentissage de représentation de séries temporelles géolocalisées (GTS) dans le contexte de la ville intelligente. Nous étudions spécifiquement la tâche de prévision du trafic routier avec un focus sur le traitement intégré des valeurs manquantes.

Abstract

Time series is a common data type that has been applied to enormous real-life applications, such as financial analysis, medical diagnosis, environmental monitoring, astronomical discovery, etc. Due to its complex structure, time series raises several challenges in their data processing and mining. The representation of time series plays a key role in data mining tasks and machine learning algorithms for time series. Yet, a few methods consider the interrelation that may exist between different time series when building the representation. Moreover, the time series mining requires considering not only the time series' characteristics in terms of data complexity but also the concrete application scenarios where the data mining task is performed to build task-specific representations.

In this thesis, we will study different time series representation approaches that can be used in various time series mining tasks, while capturing the relationships among them. We focus specifically on modeling the interrelations between different time series when building the representations, which can be the temporal relationship within each data source or the inter-variable relationship between various data sources. Accordingly, we study the time series collected from various application contexts under different forms. First, considering the temporal relationship between the observations, we learn the time series in a dynamic streaming context, i.e., time series stream, for which the time series data is continuously generated from the data source. Second, for the inter-variable relationship, we study the multivariate time series (MTS) with data collected from multiple data sources. Finally, we study the MTS in the Smart City context, when each data source is given a spatial position. The MTS then becomes a geo-located time series (GTS), for which the inter-variable relationship requires more modeling efforts with the external spatial information. Therefore, for each type of time series data collected from distinct contexts, the interrelations between the time series observations are emphasized differently, on the temporal or (and) variable axis.

Apart from the data complexity from the interrelations, we study various machine learning tasks on time series in order to validate the learned representations. The high-level learning tasks studied in this thesis consist of time series classification, semi-supervised time series learning, and time series forecasting. We show how the learned representations connect with different time series learning tasks under distinct application contexts. More importantly, we conduct the interdisciplinary study on time series by leveraging real-life challenges in machine learning tasks, which allows for improving the learning model's performance and applying more complex time series scenarios.

Concretely, for these time series learning tasks, our main research contributions are the following: (i) we propose a dynamic time series representation learning model in the streaming context, which considers both the characteristics of time

series and the challenges in data streams. We claim and demonstrate that the Shapelet, a shape-based time series feature, is the best representation in such a dynamic context; (ii) we propose a semi-supervised model for representation learning in multivariate time series (MTS). The inter-variable relationship over multiple data sources is modeled in a real-life context, where the data annotations are limited; (iii) we design a geo-located time series (GTS) representation learning model for Smart City applications. We study specifically the traffic forecasting task, with a focus on the missing-value treatment within the forecasting algorithm.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Internet of Things (IoT), which connects a variety of ubiquitous sensors, devices, and machines together, is currently gaining substantial attention in both academia and industry. Particularly, sensors positioned on various objects such as household appliances, industrial machines, vehicles, even human bodies, etc., continuously produce real-valued data, which are performed over time. A collection of these organized observations forms a *time series* (TS). Beyond the IoT context, time series can be collected from almost every scientific field, including finance, astronomy, health monitoring, images, etc.

Besides, data mining [1] is a process of discovering knowledge or extracting patterns in datasets via several mining techniques, including those in machine learning, statistics, and database systems. Data mining is an interdisciplinary domain of computer science and statistics with an overall goal of extracting information (with intelligent methods) from

a dataset and transforming the information into a comprehensible structure (i.e., *representation*) for further use, such as anomaly detection, classification, clustering, predictive analysis, etc. A good understanding of the data and the mining objective is necessary for building a reliable data mining model.

Time series mining [2] is a sub-field of data mining that has been studied for decades. With countless applications, time series mining can be applied in financial analysis, industrial monitoring, medical assistance, astronomy discovery, smart city management, environmental monitoring, etc. Time series has a complex structure with strong dependencies between the observations; it is necessary to build or learn an operable *representation* from such complex data and apply the general mining techniques accordingly for different application tasks. For instance, *time series classification* task outputs a single prediction [3] for the whole sequence, where the representation is learned from each local observation; *time series forecasting* task predicts future values from the past [4], where the learned representation is generally biased towards the recent observations in the sequence. Therefore, the time series mining requires considering not only the time series' characteristics in terms of data complexity but also the application scenarios where the data mining task is performed to build a task-specific representation.

The inter-relationship between the observations is the main characteristic of the time series data. When a time series instance is collected from one single data source, we call it *univariate time series* (UTS), otherwise *multivariate time series* (MTS). UTS and MTS are two general concepts for sequential data in various application contexts. The inter-relationship can happen on the temporal axis, e.g., the past observations in time series may affect the recent ones, or on the variables, e.g., one data source may interact with the others to generate the data at each timestamp. Considering the inter-relationship in time series is thus of upmost importance in various time series mining tasks.

Moreover, in a dynamic streaming context, the temporal relationship between the observations can be even more complex. For instance, a smart sensor continuously generates real-valued data. An off-line dataset can be easily operated and explored to build a stable data representation for further use. When the data is continuously generated, the dynamic data source leads to a *streaming* context, for which learning the temporal relationship becomes more challenging. The time series with streaming features is a common data type in real-life applications, especially in the IoT context where many sensors infinitely generate observations in real-time if there is no external intervention. An automatic data mining process on such data allows deploying the mining models without a human in the loop, thus leading to an intelligent system that can be self-adaptive to the new environment.

The inter-variable relationship is another characteristic in time series, specifically, in multivariate time series (MTS). For instance, the human activity can be detected by multiple body sensors in an MTS; the relationship between the sensors reflects the physical interactions between different body parts. Learning such inter-variable relationships from the observations helps build a reliable time series representation. Moreover, the inter-variable relationship can be learned from external variable information. For instance, the

positions of body sensors can provide additional information for learning their relationships. Similar data sources like the traffic sensors deployed in Smart City inherently integrate a spatial location for each data source. The multivariate time series with the spatial location of each variable is generally called *Geo-located Time Series* (GTS), for which the inter-variable relationship can be learned not only from the observations but also from the external information related to the data sources.

Apart from the data complexity, the actual application scenarios or the time series mining tasks require building the time series representation accordingly. Specifically, the time series mining can be related to various application scenarios with different research purposes. One classic research problem is *Time Series Classification* (TSC) which intends to predict the label of a new input TS instance (i.e., testing data) by extracting the knowledge from the collected data (i.e., training data with labels). TSC has a large range of application scenarios, including medical diagnosis, human activity recognition, industrial troubleshooting, etc. In these applications, the data requires careful labeling, conducted either during or after the data collection process. One typical example of the former one is the activity recognition task; according to the observed activity, the human can annotate the activity TS accordingly. The observation-based labeling raises several challenges for the data collection process, such as the enormous time cost, the human effort for projecting the observation into a concrete label, etc. Most of the labeling activities are conducted after the data collection process (i.e., post-labeling); the human is capable of labeling the data with (few) domain knowledge. This is quite common for the data under some specific forms, such as images or texts, for which the labels can be easily inferred or identified. However, time series require a more significant labeling effort. The real-valued sequence does not provide explicit information related to its labeling. An expert with specific domain knowledge is necessary to conduct the post-labeling on this type of data. This limits the availability of labeled time series for the purpose of supervised learning.

Overall, the time series data can be collected from various data sources and can be processed with different mining objectives. With enormous real-life applications, time series mining has attracted widespread attention from researchers of multiple domains. The interdisciplinary study on time series is usually envisaged to improve the model's performance or broaden the application scenarios. For instance, we can consider the time series in the streaming context, with label constraint, or with a complex inter-relationship between the observations under some specific application contexts, etc. Learning a task-specific representation of these data raises different challenges. A thorough exploration of the data characteristics and a deep understanding of the application context are critical for building a reliable mining model on time series.

1.2 General Objectives

As described in the previous section, representation learning on time series covers a wide range of downstream learning tasks; there are various real-life challenges on each. Therefore, we divide the representation learning task into three main objectives:

1. **Supervised Learning in streaming context:** As the data are continuously generated in the real world, the first objective is to learn the dynamic representations of Time Series in the streaming context, which capture the temporal relationship between the observations. Compared to the static representation learning from a constant Time Series database, dynamic representation learning provides the expert and the user an adaptive model to learn from the new coming data without re-training the model from scratch. In addition, the adaptive model should be desgined with a high interpretability, which is two-fold: a) for the expert, the learning process should be explainable and can be traced for any learning issues; b) for the user, the learned dynamic representations should be interpretable in the streaming context.

2. **Learning weakly labeled Multivariate Time Series:** Apart from the temporal relationships, Multivariate Time Series (MTS) has a strong inter-variable relationship. The post-labeling of Multivariate Time Series (MTS) is much more costly than classic data (e.g., images, text, etc.) due to the low interpretability over the real-valued sequence. Therefore, most MTS data, such as sensor readings, are labeled during the data collection process. Nevertheless, the label shortage is a practical constraint for learning from MTS. The second objective is to learn the MTS representation in a semi-supervised manner, preserving the interpretability of the learned representations to the users. Moreover, We should explore the difference for learning representations from Univariate Time Series and Multivariate Time Series (i.e., inter-variable relationship).

3. **Geo-located Time Series analysis:** The third objective is to consider complex Time Series in the Smart City context. Specifically, we aim to learn the inter-variable relationship with the help of the spatial locations of the variables (e.g., geo-located sensors). Moreover, we need to consider the real-world challenge of missing values for learning the representations from the geo-located Multivariate Time Series. One of the application scenarios is *Traffic Forecasting*, for which the sensors are deployed on the road network providing extra spatial information for learning the representations to forecast the traffic situation in the future. In reality, the traffic data usually contains missing values due to sensor or communication errors. The Spatio-temporal feature in traffic data leads to a complex missing-value context: 1) in temporal axis, the values can be randomly or consecutively missing; 2) in spatial axis, the missing values can happen on one single sensor or multiple sensors simultaneously. A model considering the complex missing-value context can

give reliable results when forecasting on Geo-located Multivariate Time Series.

1.3 Contributions

Therefore, this thesis tackles the three objectives mentioned above. We first present the related state-of-the-art approaches for each of these goals. We then show our contributions in each of the three following topics: (i) We first consider the case of dynamic representation learning on Time Series Stream, which corresponds to the objective of supervised learning in streaming context; (ii) We then explore the case of semi-supervised representation learning on Multivariate Time Series, regarding the objective of learning weakly labeled Multivariate Time Series; (iii) Finally, we learn the representations for forecasting task on the geo-located Multivariate Time Series with missing values, which is related to the objective of geo-located Time Series analysis.

1.3.1 Dynamic Representation Learning on Time Series Stream

We start by introducing the contribution related to the Supervised Time Series Representation Learning task in the streaming context. In the real world, the time series data are usually generated continuously; for instance, the city sensors constantly monitor the city's situations (e.g., traffic, air pollution, etc.), thus generating infinite time series with streaming characteristics. The database can be continuously enriched with newly arrived time series samples, for which the temporal relationships may evolve over time. Considering the supervised features in the dynamic time series, an incremental and adaptive representation learning model allows the experts and users to explore the system's evolution without extra training effort, thus improving both the research and economic efficiency.

1.3.1.1 Limitation of Current Approaches

Most time series classification (TSC) approaches are biased towards learning from an offline time series dataset, with the assumption that data instances are independently and identically distributed (i.i.d) within a particular concept, but rarely consider the streaming context, where a gradual change of the concept happens along with the input of TS stream, that is *Concept Drift* [5]. For instance, the most accurate ensemble classifiers [6, 7] for time series are not good options in the streaming context due to their complex architecture. Lazy classifiers on time series such as Nearest Neighbor (1-NN) [8], and dictionary-based approaches [9] are applicable for streaming context. However, every input instance will be considered to adjust the inner concept, which potentially requires an ample buffer space and will bring a considerable computation cost. Recent Deep Neural Network (DNN) approaches [3, 10, 11, 12] on TSC tasks are capable of tuning the model incrementally but always stay in an awkward position for the lack of explainability, which is required by

domains like healthcare where questions of accountability and transparency are particularly important.

1.3.1.2 SMAP

First, we propose SMAP, a scalable approach based on Matrix Profile [13] for extracting Shapelet features from large Time Series. SMAP is applicable to certain fields in Big Data context, where the TS features and their extraction process should be interpretable. It improves the state of the art solutions by proposing a scalable and highly efficient method to classify TS based on characteristic subsequences (i.e., shapelets). This work is biased towards raw time series processing which has a higher accuracy performance but a relatively high time complexity [14]. Traditional TSC algorithms on raw TS data are not applicable for big data context due to their low scalability. Unlike some hardware-based implementations, such as using GPUs to accelerate the similarity calculation [15], we focus on the scalability of TSC based on shapelet extraction with a distributed framework named *Spark*. Here are our contributions in this work:

- We propose a novel method to assess the importance of shapelets in batches.
- We introduce a scalable engine to extract the shapelets with *Spark* framework.
- Based on the scalable engine, we propose an optimization strategy to speed up the shapelets extraction.

1.3.1.3 ISMAP

Second, considering the dynamic temporal relationship between time series instances in the streaming context, we propose ISMAP, an incremental and adaptive approach for learning representations from Time Series Stream. ISMAP fills the gap between Time Series Classification and data streams processing, where TS features and models can be updated with consideration of Concept Drift, without retraining the model from scratch. The incremental version of SMAP under Spark framework allows us to further explore the *Test-then-Train* strategy, to evaluate the learning model constantly on newly input instances, then update the model regarding the evaluation result. The cached information under the old concept will be eliminated gradually by an elastic caching mechanism, which deals with the challenge of infinite streaming instances.

The main contributions of this work are the following:

- **Scalability:** Our algorithm conserves the scalability of Shapelet Extraction [16] in the streaming context, which is always parallelizable in a remote Spark cluster with a minimum communication cost between distributed nodes.

- **Shapelet Evaluation:** We propose a novel strategy to evaluate Shapelet, which shows the first attempt of transferring the techniques in time series community to data stream community.

- **Test-then-Train**: The novel strategy, not only accelerates the incremental Shapelet extraction in a stable-concept context, but also helps with detecting Concept Drift in the streaming context by *Test-then-Train* technique.

- **Explainability:** The algorithm shows not only interpretability of extracted features (i.e., Shapelets), but also a strong explainability of Shapelet Evolution in dynamic source context.

- **Traceability:** The system allows to trace and explain the learning model at different time ticks, which gives us a possibility to supervise the system and back up the historical features.

1.3.2 Semi-supervised Learning on Multivariate Time Series

We now show the contributions related to the semi-supervised learning task on Multivariate Time Series (MTS). On the one hand, the label shortage is a real-life problem for learning from any type of data. This problem becomes even more critical in the MTS context due to the low interpretability over the real-valued sequence, for which more efforts are required for the data annotations compared to the classic data (e.g., images, text, etc.). On the other hand, the MTS has a complex data structure on both temporal relationship and inter-variable relationship, which requires more processing techniques than the Univariate Time Series (UTS).

1.3.2.1 Limitation of Current Approaches

Recent studies usually learn the representations on Multivariate Time Series in an unsupervised manner. For instance, [17] uses triplet loss to form the embedding space; then even an SVM classifier is powerful enough on the learned representation [18]. However, existing techniques suffer from three major issues. First, the relationships between the MTS variables are generally computed on the entire 1-D series, ignoring the fact that the local relationships at the sub-sequence level may evolve in the dynamic sequence, that is *spatial dynamics*. Second, the representation learned in a purely unsupervised manner depends mainly on the loss function selection. As no label information is utilized to learn the representation [17], there is a risk that it deviates from the true features, thus affecting the classifier performance. Third, they rather employ deep learning as a blind box and do not focus on the interpretability of the learned representation.

1.3.2.2 SMATE

To handle both the label shortage problem and the complex relationships between MTS observations, we propose SMATE, **S**emi-supervised Spatio-temporal representation learning on **M**ultiv**A**riate **T**ime S**E**ries. The auto-encoder-based structure allows mapping the MTS samples from raw features space $\mathcal{X}$ to low dimensional embedding space $\mathcal{H}$. A Spatial Modeling Block combined with a multi-layer convolutional network captures the spatial dynamics, whereas a GRU-based structure extracts the temporal dynamic features. Thereby, SMATE is capable of compressing the essential Spatio-temporal characteristics of MTS samples into low-dimensional embeddings. On top of this embedding space $\mathcal{H}$, we propose a semi-supervised three-step regularization process to bring the model to learn class-separable representations, where both the labeled and unlabeled samples contribute to the model's optimization. This regularization process comes with the capability of visualization at each step, making SMATE interpretable.

We summarize the main contributions in this work as follows:

- **Spatio-temporal dynamic features in MTS:** We claim and demonstrate that the temporal dependency and the evolution of the variable interactions (*spatial dynamics)* are important for building a reliable MTS embedding.
- **Weak supervision on learning representations:** With limited labeled data, SMATE can learn reliable class-separable MTS representations for downstream tasks, such as MTS classification (MTSC).
- **Interpretable MTS embedding learning:** SMATE allows for visual interpretability, not only from the class-separable representations but also in each step of the semi-supervised regularization process.
- **Extensive experiments on the MTS datasets:** The experiments are carried out on various MTS datasets from different application domains. The detailed evaluation with 13 supervised baselines and four semi-supervised works is provided, which shows the effectiveness and the efficiency of SMATE over state of the art.

1.3.3 Geo-located Multivariate Time Series Forecasting with Missing Values

We now describe the contributions related to the forecasting task on the geo-located Multivariate Time Series (MTS) with missing values. The geo-located MTS provides external spatial information for the data sources (i.e., variables) with primary applications in the Smart City context. On the one hand, a thorough exploration of the spatial information is able to help us better capture the inter-variable relationships, thus improving the model performance. On the other hand, the geo-located MTS is usually accompanied by complex missing values. Considering theses two characteristics, the learned model can be applied

to various real-life applications, not limited to Traffic Forecasting, but also on crowd flow forecasting [19], weather and air pollution forecasting [20], etc.

1.3.3.1 Limitation of Current Approaches

Recent work [21, 22, 23, 24, 25] tend to jointly consider the missing values and the forecasting modeling during the training step (i.e., *one-step processing*) and declared a better performance than the two-step processing. However, the above-mentioned work suffers from three major issues. First, the missing and zero values are usually considered indifferent, leading to unnecessary, even harmful data imputations, thus contradicting the raw data information. Second, most of the work [21, 22, 24, 25] considers missing values from the temporal aspect, ignoring the rich information from the spatial perspective. Third, they are generally designed for processing the missing values in some basic scenarios, such as random missing or temporal block missing, but lack of power for the complex scenarios. In the real-world, the missing values in traffic data occur on both long-range (e.g., *device power-off*) and short-range (e.g. *device errors*) settings, on partial (e.g., *local sensor errors*) and entire transportation network (e.g., *control center errors*). The complex scenarios require a holistic approach for handling various types of missing values together.

1.3.3.2 GCN-M

Therefore, to handle both the Spatio-temporal patterns and complex missing-value scenarios in traffic data, we propose GCN-M, **G**raph **C**onvolutional **N**etworks for Traffic Forecasting with **M**issing Values. The graph neural network-based structure allows jointly modeling the Spatio-temporal patterns and the missing values in one-step processing. We construct local statistic features from spatial and temporal perspectives for handling short-range missing values, which is further enhanced by a memory module to extract global historical features for processing long-range missing blocks. The combined local-global features allow not only identifying the missing measures from the inherent zero values but also enriching the traffic embeddings, thus generating dynamic traffic graphs to model the dynamic spatial interactions between traffic nodes. The missing values on a partial and entire network can then be considered from spatial and temporal perspectives.

We summarize the contributions in this work as follows:

- **Complex missing value modeling**: We study the complex scenario where missing traffic values occur on both short & long ranges and partial & entire transportation networks.
- **Spatio-temporal memory module**: We propose a memory module that can be used by GCN-M to learn both local Spatio-temporal features and global historical patterns in traffic data for handling the complex missing values.

- **Dynamic graph modeling**: We propose a dynamic graph convolution module that models the dynamic spatial interactions. The dynamic graph is characterized by the learned local-global features at each timestamp, which not only offset the missing values' impact but also help learn the graph.

- **Joint model optimization**: We jointly model the Spatio-temporal patterns and missing values in one-step processing, which allows processing missing values specifically for traffic forecasting tasks, thus bringing better model performance than two-step processing.

- **Extensive experiments on real-life data**: The experiments are carried out on two real-life traffic datasets. We provide detailed evaluations with 12 baselines, which show the effectiveness of GCN-M over the state-of-the-art.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows.

In **Chapter 2**, we review the state of the art and formulate the research problems. First, we introduce basic definitions and taxonomy related to the representation learning on Time Series mining (Section 2.1). Then, we review the state-of-the-art approaches of learning Time Series Representations for both classification (Section 2.2) and forecasting (Section 2.3).

In **Chapter 3**, we discuss the supervised representation learning task on Time Series Stream. First, we show the positioning of our work in state-of-the-art and the limitations of previous work related to our research problem. Then, we introduce two approaches for supervised representation learning on large time series. We first show SMAP, an off-line approach for extracting Shapelet features from large Time Series in a distributed manner. We then describe ISMAP, an online version of SMAP which allows learning the Time Series Representation in a streaming context. Finally, we show the experimental results on real-life datasets under the streaming context.

In **Chapter 4**, we discuss the semi-supervised representation learning task on Multivariate Time Series. We first describe the limitations of the current state-of-the-art approaches when learning representations from Multivariate Time Series. Then, we introduce SMATE (Section 4.4), a semi-supervised Spatio-temporal representation learning model on Multivariate Time Series. We finally show an extensive experimental analysis (Section 4.5) on MTS datasets from various application domains under both supervised and semi-supervised settings.

In **Chapter 5**, we discuss the forecasting task on the geo-located Multivariate Time Series (MTS) with missing values. Specifically, we adopt the *Traffic Forecasting* scenario for the learning task. First, we show the limitation of previous Traffic Forecasting work when

considering both Spatio-temporal structure and the complex missing-value scenario. Then, we describe GCN-M (Section 5.4), a graph convolutional network-based model for traffic forecasting with missing values. Finally, we show the experimental results on real-life traffic data and compare them with the state-of-the-art approaches.

1.5 List of Publications

- J. Zuo, K. Zeitouni, Y. Taher, S. G. Rodriguez: Graph Convolutional Networks for Traffic Forecasting with Missing Values. **[Under Review]**
- H. El Hafyani, M. Abboud, J. Zuo, K. Zeitouni and Y. Taher: Learning the Micro-environment from Rich Trajectories in the context of Mobile Crowd Sensing -Application to Air Quality Monitoring. **[Under Review]**
- J. Zuo, K. Zeitouni, Y. Taher: Semi-supervised Spatio-Temporal Representation Learning on Multivariate Time Series, IEEE ICDM, Virtual Conference (2021)
- H. El Hafyani, M. Abboud, J. Zuo, K. Zeitouni and Y. Taher: Tell Me What Air You Breath, I Tell You Where You Are, demonstration track, SSTD, Virtual Conference (2021)
- M. Abboud, H. El Hafyani, J. Zuo, K. Zeitouni and Y. Taher: Micro-environment Recognition in the context of Environmental Crowdsensing, BMDA@EDBT, Virtual Conference (2021)
- J. Zuo, K. Zeitouni, Y. Taher: Incremental and Adaptive Feature Exploration over Time Series Stream, IEEE BigData, Los Angeles, USA (2019)
- J. Zuo, K. Zeitouni, Y. Taher: ISETS: Incremental Shapelet Extraction from Time Series Stream, demonstration track, ECML-PKDD, Würzburg, Germany (2019)
- J. Zuo, K. Zeitouni, Y. Taher: Exploring Interpretable Features for Large Time Series with SE4TeC, demonstration track, EDBT, Libon, Portugal (2019)
- J. Zuo: Time Series meet Data Streams: Perspectives of the Interdisciplinary Collision and Applications, PhD symposium, BDA, Lyon, France (2019)

Chapter 2

State of the art

Contents

2.1 Time Series Data Mining

Time Series Mining is a classical research problem in the data mining community, which has been studied for decades. Due to the complex data structure and board application domains of the time series, various mining activities [26] have been conducted on top of

the time series, such as classification, clustering and forecasting, studied by the machine learning community, or indexing, segmentation considered by the database community, etc. This chapter introduces the necessary background for mining time series data from the perspective of the representation learning [27]. We start by introducing briefly the basic definitions related to time series mining. Then, we pick two downstream learning tasks (classification, forecasting) and present the state-of-the-art work.

2.1.1 Definition and taxonomy

Definition 2.1. (Time series). A time series (TS) $\mathbf{x} \in \mathcal{R}^{T \times M}$ is a sequence of real-valued vectors: $\mathbf{x}$=(x_1, x_2, ..., x_t, ..., x_T), where $x_t \in \mathcal{R}^M$, M is the variable number. When $M = 1$, we call it *univariate time series* (UTS), otherwise we call it *multivariate time series* (MTS).

Definition 2.2. (Subsequence). Given a time series $\mathbf{x}$ of length T, a *subsequence* $\boldsymbol{x}_{t,m}$ of $\mathbf{x}$ is a continuous subset of values from $\mathbf{x}$ of length $m \leq T$ starting from index t: $\mathbf{x}_{t,m} = (x_t, x_{t+1}, ..., x_{t+m-1})$, where $t \in [0, n - m + 1]$.

Definition 2.3. (Time series representation). Given a time series $\mathbf{x} \in \mathcal{R}^{T \times M}$, a time series representation $\mathbf{r} \in \mathcal{R}^{T' \times M'}$ is a summarised (i.e., $T' \times M' < T \times M$) feature set of the original time series $\mathbf{x}$, which closely approximates $\mathbf{x}$.

Definition 2.4. (Time series similarity measure). The *similarity measure* $D(\mathbf{x}, \mathbf{x}')$ of time series $\mathbf{x}$ and $\mathbf{x}$' is a function taking two time series as inputs and returning the *distance* between these series.

Time series data differs from static data in a way that the ordering of the data attribute in time series data is critical in finding the best discriminating features.

The purpose of time series mining is to extract all meaningful knowledge from complex temporal data. As shown in Figure 2.1, according to the mining activities on time series, we roughly organize the time series mining into two categories: (i) Learning-based activities, where the typical machine learning tasks are conducted to explore the valuable knowledge in time series; (ii) Non-learning-based activities, where the high-level mining tasks specifically designed for time series data are studied. These tasks are usually defined as theoretical objectives, and the actual applications may cover multiple tasks simultaneously. In other words, the intersections between the mining activities are practically common depending on the concrete context. For instance, the *Anomaly Detection* in time series is defined differently according to the context: when the anomaly's nature is known in advance, e.g., a *subsequence*, then it is related to a *Indexing-Query* problem in the time series database; when the anomaly is unknown, one consensus [28] is to consider the rare motif in the sequence as an anomaly, then it is related to the *Motif Discovery* task. In the rest of this section, we describe the mining activities briefly within each category.

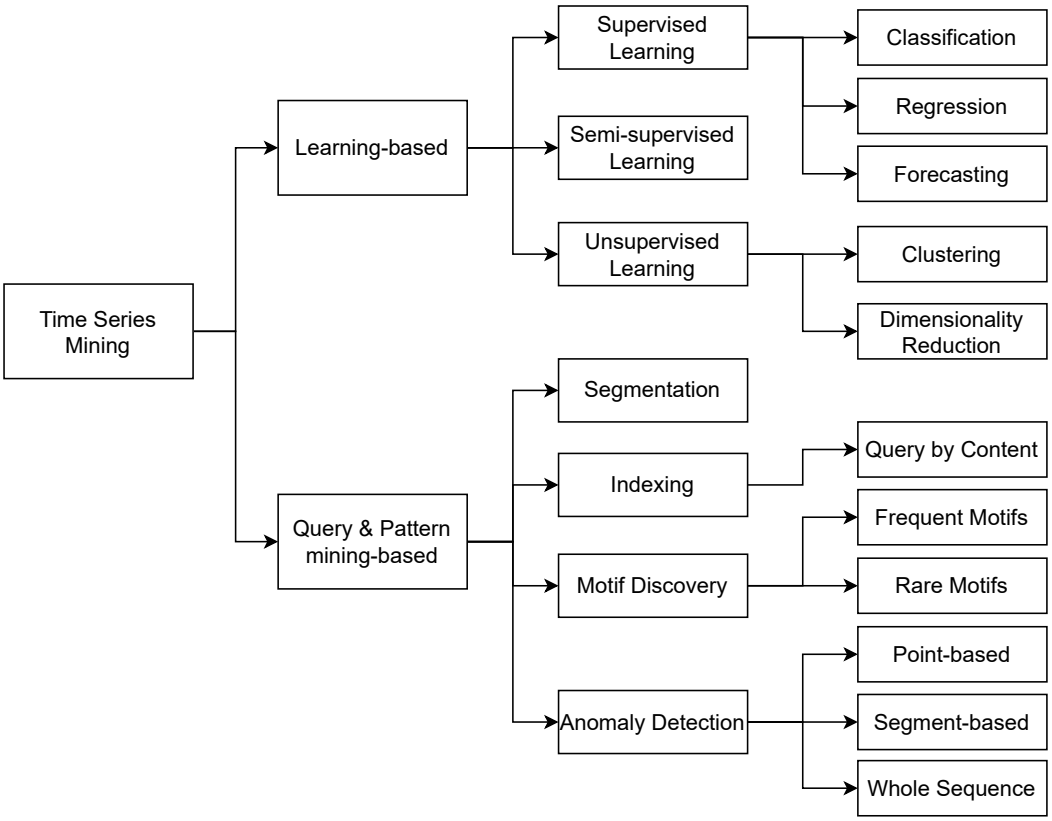


Figure 2.1: Taxonomy of time series mining activities

2.1.1.1 Learning-based Time Series Mining

Definition 2.5. (Time series classification). Given a training set $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with N time series $\mathbf{x}^{(i)}$ and their associated label (target variable) $y^{(i)}$, where $y^{(i)}$ is a categorical value, we aim to learn a classification model f such that $f(\mathbf{x}^{(i)}) = y^{(i)}$, in order to predict correctly the labels of new testing time series.

Definition 2.6. (Time series regression). Given a training set $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with N time series $\mathbf{x}^{(i)}$ and their associated label (target variable) $y^{(i)}$, where $y^{(i)}$ is a continues scalar value, we aim to learn a regression model f such that $f(\mathbf{x}^{(i)}) \approx y^{(i)}$, in order to predict precisely the labels of new testing time series.

Definition 2.7. (Time series forecasting). Given a training set $\mathcal{D} = \{\mathbf{x}_{1:T}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with N time series $\mathbf{x}_{1:T}^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_t^{(i)}, ..., x_T^{(i)})$ and their associated label (target variable) $y^{(i)} = (x_{T+1}^{(i)}, ..., x_{T+\tau}^{(i)})$, we aim to learn a forecasting model f such that $f(\mathbf{x}^{(i)}) \approx y^{(i)}$, in order to forecast precisely the future values of a new testing time series. The time point $T+1$ is referred to as *forecast start time* and $\tau \in \mathcal{N}_{>0}$ is the *forecast horizon*. When $\tau = 1$, we call it *single-step forecasting*, otherwise we call it *multi-step forecasting*.

Time series classification, regression, and forecasting are supervised learning tasks that learn the relationship between a target variable and an input time series representation. The main difference between the learning tasks is related to the target variables. For the *classification* tasks, we predict a categorical value for a data instance that categorizes the data into some finite categories; for the *regression* tasks, the model outputs a continuous value; for the *forecasting* tasks, the model predicts a value or a sequence in the future. We should note that these learning tasks are mutually convertible. For instance, the *Regression* tasks become *Classification* tasks when the predicted values are discretized into some finite labels for the data. Similarly, the *Forecasting* tasks can become *Regression* tasks when we set the *forecast horizon* to 1, i.e., *single-step forecasting*.

Definition 2.8. (Semi-supervised time series learning). Given a weakly labeled TS dataset $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$ which contains both labeled and unlabeled TS samples:

$$\mathcal{D}_l = \{\mathbf{x}^{(i)}, y_i\}_{i=1}^{N*r},\ \mathcal{D}_u = \{\hat{\mathbf{x}}^{(i)}\}_{i=1}^{N*(1-r)}$$

where r $(0 \leq r \leq 1)$ indicates the *ratio* of the labeled samples in $\mathcal{D}$ of size N, y_i is the annotation of the labeled instance $\mathbf{x}^{(i)}$, the semi-supervised time series learning aims at training a model to predict successfully the label of a testing time series, adopting the supervised training from $\mathcal{D}_l$ and further unsupervised adjustment/optimization from $\mathcal{D}_u$.

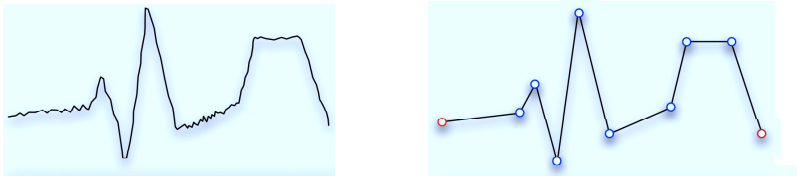
Definition 2.9. (Unsupervised time series learning). Given an input set $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ composed of N time series $\mathbf{x}^{(i)}$ without any target variables, the *unsupervised learning* on time series aims to build representations of the input $\mathcal{D}$ that can be used for decision making, predicting future inputs, etc.

Two classic examples of unsupervised time series learning are *time series clustering* [29] and *time series dimensionality reduction* [30], in which a time series representation [31] is learned to conserve the primary features in time series.

2.1.1.2 Query & Pattern mining-based Time Series Mining

Unlike the learning-based activities, the Query & Pattern mining on time series are generally high-level mining tasks that cover various specific techniques for processing time series data.

Definition 2.10. (Time series segmentation). Given a Time Series $\mathbf{x}$=(x_1, x_2, ..., x_t, ..., x_T), *time series segmentation* aims to construct a model f which divided $\mathbf{x}$ into a sequence of K discrete segments $f(\mathbf{x})$= $\{\mathbf{r}^{(i)}\}_{i=1}^{K}$ such that $f(\mathbf{x})$ closely approximates $\mathbf{x}$, where $\mathbf{r}^{(i)}$ is a linear segment.



(a) the raw time series **(b)** the segmented time series

Figure 2.2: (Figure taken from [2]) Example of the time series segmentation.

As shown in Figure 2.2, the time series segmentation generates an accurate approximation of the raw time series. The segmentation operation can be considered either as a discretization problem [32] or as a preprocessing step for various mining tasks, such as *time series classification* [33], *time series clustering* [34], etc. The most common approach for segmenting time series is Piecewise Linear Approximation (PLA) [35]. The main idea behind PLA is first to split the series into most representative segments, then to fit a polynomial model for each segment. A good review on the splitting and fitting techniques based on PLA can be found in [32].

Definition 2.11. (Time series indexing). The time series indexing aims to build a scheme that efficiently organizes time series data for quick retrieval/query in large time series databases.

Most time series indexing approaches involve a dimensionality reduction process in order to index the low-dimensional time series representations (a.k.a., summarization). A classical review of time series representations for dimensionality reduction can be found in [31], which suggests that the querying speed (i.e., efficiency) and the quality of querying results (i.e., effectiveness) are two main issues when building the indexing scheme. To support efficient querying (a.k.a., similarity search) in a time series database, the design of distance measure and summarization techniques is critical. A detailed review for handling efficient time series querying can be found in [36].

Definition 2.12. (Motif Discovery). Given a *long* Time Series $\mathbf{x}$ of length T, the *Time series motif* discovery is to extract the approximately repeated subsequence from $\mathbf{x}$.

Originally defined in [37], time series motifs are *typical non-overlapping subsequences*, which can be frequently repeated, such as the common heartbeat ECG recordings (less than once per second) [38] of a specific patient, or infrequently repeated, such as the repeating earthquake source [39] in Sonoma County, California, that has a frequency of about once per 13.6 years.

Definition 2.13. (Time series anomaly detection). A *time series anomaly* is an observation or a sequence of observations that deviates remarkably from the general distribution of data. The set of anomalies forms a tiny part of the dataset. Given a Time Series $\mathbf{x}$

of length T, the anomaly detection aims to identify the location of the abnormal observation (*point-based anomaly*) or sequence of observations (*segment-based/whole sequence anomaly)* in $\mathbf{x}$.

There are two main characteristics of anomalies: (i) The distribution of the anomalies deviates remarkably from the general distribution of the data; (ii) The big majority of the dataset consists of normal data points. The anomalies form only a tiny part of the dataset. The time series anomaly detection is a high-level mining activity covering various mining tasks depending on the techniques and concrete context. For instance, knowing a-priori which kind of anomaly the data light contain, converts the detection task as an *Indexing-Query* problem in the time series database; In contrast, if the anomaly is unknown, it can be related to the *Rare Motif Discovery* task [40], *Time Series Forecasting* task [41], or *Time Series Clustering* task [42] after segmenting the time series [32].

2.1.2 Representation Learning on Time Series

Definition 2.14. (Time series representation learning). Also known as time series feature learning, given $\mathbf{x} \in \mathcal{R}^{T \times M}$, it aims to build a model f, to learn a representation $\mathbf{r} = f(\mathbf{x}) \in \mathcal{R}^{T' \times M'}$ $(T' \times M' < T \times M)$ of the time series $\mathbf{x}$, which conserves the key characteristics of $\mathbf{x}$ and can be applied to one or multiple downstream learning tasks, such as classification, regression, forecasting, clustering, etc.

Time series has a *high dimensionality*, the observation at each timestamp can be considered as an individual dimension. Therefore, the data mining algorithms that work directly on raw time series can be computationally expensive and storage costly. The main motivation of constructing or learning a data representation is to emphasize the essential characteristics of the data in a concise way. With an appropriate representation, the mining tasks obtain improvements not only on the model efficiency but also on the model's effectiveness [31]. Additional benefits obtained are efficient data/model storage [43], removal of random noise [44], and the possibility of processing multi-model data under a unified view [27]. For instance, the time series representation integrating the key characteristics (e.g., temporal ordering features) allows applying the algorithms designed for static data for various downstream time series mining tasks (e.g., *classification*, *clustering*, etc.). The data representation can be considered as a transformation of the complex input data into a unified processing unit. These basic properties lead to the following requirements for any time series representation techniques: (i) the data dimensionality should be significantly reduced; (ii) the essential time series characteristics should be conserved; (iii) the representation should be computationally efficient; (iv) the representation should be insensible to noise or be capable of handling noisy time series.

The time series representation is a primary research issue in time series mining. There are various representation techniques designed for different purposes in the literature. We organise the representation techniques into three basic categories: *transformation-based*

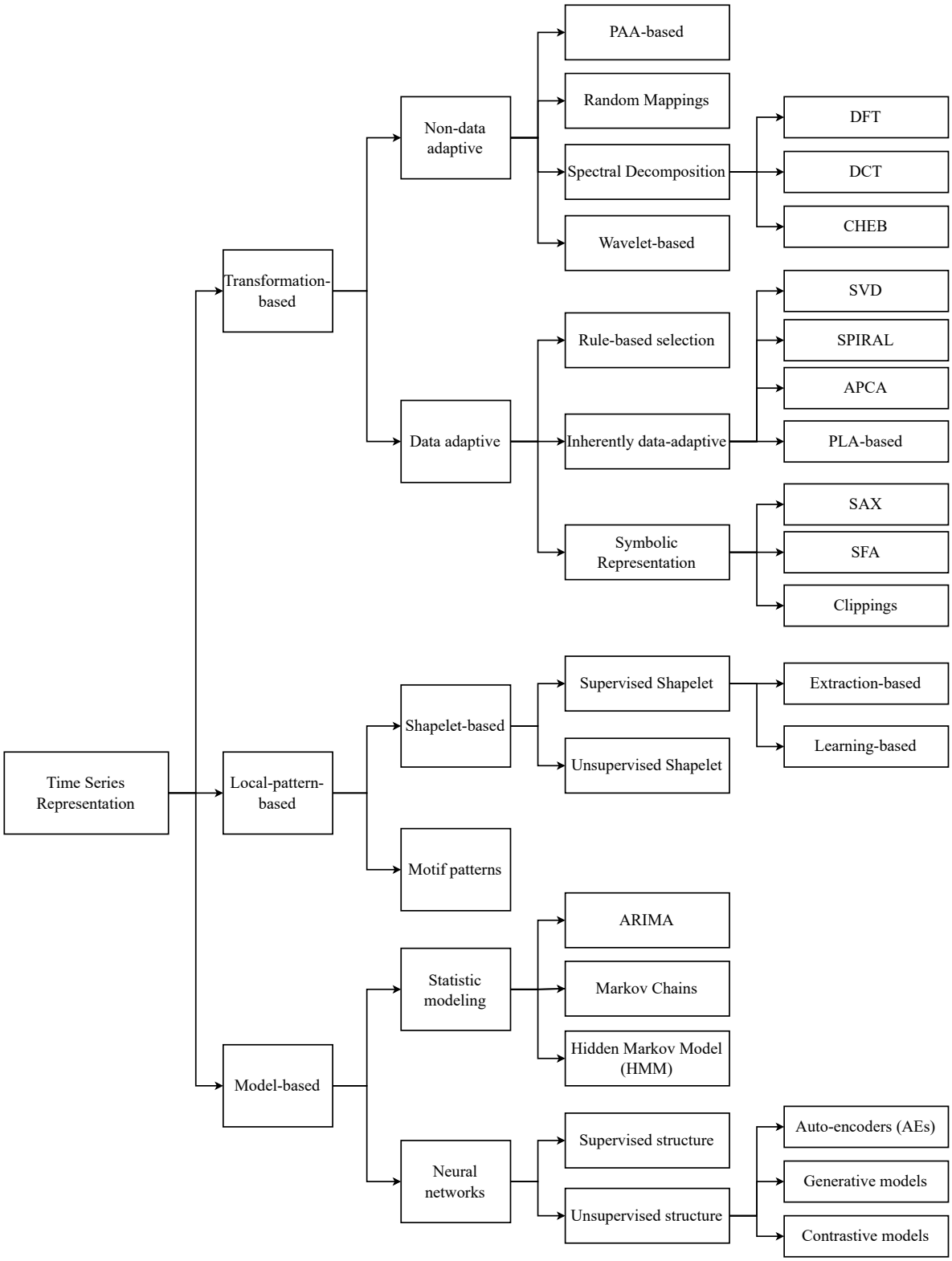


Figure 2.3: Taxonomy of time series representation approaches

representations, *local pattern-based* representations, and *model-based* representations. Figure 2.3 shows a taxonomy for building a time series representation.

2.1.2.1 Transformation-based Representations

In transformation-based representations, a set of *rules* are applied to transform the entire time series into a low-dimensional representation, which roughly conserve the global characteristics of the time series. According to [31], the *rules*-based representations can be divided into two basic categories: *Non-Data Adaptive* and *Data Adaptive*.

Non-Data Adaptive. The non-data adaptive representation applies identical parameters or operations for transforming every time series in the database regardless of its nature.

The most commonly used techniques are based on the Piecewise Aggregate Approximation (PAA) [45, 46] (also called *Piecewise Constant Approximation* in [47]), which represent the time series by using the mean values of consecutive fixed-length segments. Authors in [48] propose an extended version called *Adaptive Piecewise Constant Approximation* (APCA), in which the length of each segment is adaptive to the shape of the time series. In addition, [49] proposes *Multi-resolution* PAA which constructs the representation at various resolution levels. Instead of using mean values, [50] adopts the segmented sum of variation (SSV) to represent each segment, while [51] proposes extracting and concatenating the amplitude-levelwise local features (ALF) of each local segments to represent the whole series. Furthermore, a bit-level approximation is proposed by [52, 53], which uses a bit to represent each data point. Authors in [54, 55, 56] adopt a set of statistic features as the time series representations, which can be extracted from each segment or the entire series.

Random Projection [57] is another technique for transforming time series. The main idea is to apply k random vectors on the time series to compute the convolution products. In this way, the time series are projected into k sub-spaces leading to a larger feature space for downstream learning tasks, e.g., *classification*, *clustering*, etc. This technique has been adopted by [58] for doing statistic queries in streaming time series, and by [12, 59] for classifying time series with thousands of fixed convolutional kernels. Similarly, the kernel methods such as the Random Warping Series (RWS) [18], Wasserstein Time Series Kernel (WTK) [60], and Generic RepresentAtIon Learning (GRAIL) [61] have great promise for learning time series features with non-linear models by implicitly projecting the time series into rich-feature representations.

The methods, as mentioned above, transform time series in *time domain* directly. Transforming time series in the *spectral domain* is another large family of approaches. *Spectral decomposition* allows decomposing the time series with various basis signals and corresponding coefficients. For instance, the Discrete Fourier Transforms (DFT) [62] adopts a sin and cosine function basis [63], the Discrete Cosine Transform (DCT) [64] uses only a cosine basis, and [65] applies Chebyshev polynomials (CHEB) as a basis. These methods

consider the global characteristics of the time series, which are applicable for stationary time series. The wavelet-based approaches are capable of conducting time-frequency analysis which allows extracting the local frequency characteristics of different intervals in non-stationary time series. For instance, the Discrete Wavelet Transform (DWT) has been found to be effective in replacing DFT [66], which uses a scaled and shifted version of a basic *wavelet* function. In addition, various wavelet functions have been investigated by researchers, such as Haar [67], Daubechies [68] or Coiflets [69]. Authors in [70] provide a detailed comparison between different wavelet functions. Besides, [71] compares DFT and DWT for the similarity search task in the time series database. Finally, a combination of DFT and DWT is applied in [72] to extract both local and global frequency characteristics in time series.

All the non-data adaptive methods apply the same function or operation with identical parameters to transform the time series in the database. Authors in [31] show that *these methods are generally less efficient than the data adaptive approaches as they do not adapt to each individual data.*

Data Adaptive. In data adaptive approaches, the transforming function or operation parameters are dynamic according to the time series instance.

First, when considering an additional data-sensitive selection step, almost all non-data adaptive methods can become data adaptive. For instance, the data-adaptive version of DFT [73] and DWT [74] allows selecting the best Fourier or Wavelet coefficients on the individual time series samples. As for the data-adaptive methods based on PAA, authors in [75] propose Piecewise Vector Quantized Approximation (PVQA), which partitions each sequence into equal-length segments and uses vector quantization to represent each segment by the closest codeword from a codebook of key-sequences. Based on PVQA, the Multiresolution Vector Quantized Approximation (MVQA) [76] keeps both local and global information about the original time series in a hierarchical mechanism, processing the original time series at multiple resolutions.

Second, some methods are inherently data adaptive. For instance, the Singular Value Decomposition (SVD) proposed in [64] considers the time series database as a huge data matrix that can be decomposed into low-dimensional matrices. However, SVD is computational and storage costly [31] thus untenable for large datasets. Similarly, a recent work named SPIRAL [77] takes the entire dataset of time series with various lengths and converts it into an instance-feature matrix, where the feature is represented by the similarity between the time series instances. Instead of considering the interrelation between time series instances as the representation, another set of studies works directly on each individual time series. The Adaptive Piecewise Constant Approximation (APCA) [48] approximates each time series by a set of constant value segments of *varying lengths* such that their individual reconstruction errors are minimal. The Piecewise Linear Approximation (PLA) [35] is a widely used representation for the time series segmentation task. Each linear segment in PLA is represented by a set of polynomial coefficients and can be obtained either by interpolation [78] or regression [79]. A large number of PLA variants have

been proposed for different purposes. For instance, [80] extends PLA to a multi-resolution setting; [81] proposes an Indexable PLA to accelerate the time series indexing process; [82] extends PLA into the context of streaming time series, which prioritizes the recent queries compared to the old ones.

Finally, instead of discretizing the time series into numeric segments, it is also possible to transform the sequence into a set of symbols. The symbolic representations would allow researchers to avail of the wealth of data structures and algorithms from the text processing and bioinformatics communities [83]. For instance, [84] encodes the shape of time series into an alphabet of characters and then to treat them as text; [79] transforms time series into symbol strings using change ratio between contiguous data points which allow building a suffix tree to index all suffixes of the symbol strings. An early review of the symbolization or symbolic time series analysis can be found in [85]. However, these methods did not define a distance measure in the symbolic space, which is correlated with the distance measures defined on the original time series, thus providing no lower bounding guarantee for efficiently manipulating the symbolic representation. The Symbolic Aggregate approXimation (SAX) [86] first transforms the data into the Piecewise Aggregate Approximation (PAA) representation [45] and then symbolizes the PAA representation into a discrete string. In this way, the symbolic distance measure defined in SAX would allow lower bounding the PAA distance. Various SAX extensions have been proposed in the literature. For instance, apart from considering the mean values of the time series segments when building PAA, the extended SAX (ESAX) [87] uses additional two new points, that is, max and min points, for segment approximation; 1d-SAX [88] integrates the slope value of the segment for modeling the time series trend, which is followed by [89] for improving the trend distance measure. Instead of using PAA representation as the first transformation step, the Symbolic Fourier Approximation (SFA) [90] symbolizes the time series over its DFT representations [62]. Recent work [9, 44] adopt the "*bag-of-words*" concept from the text mining community to represent the symbolized time series, two typical representations are Bag-of-SAX-Symbols [9, 91] and Bag-of-SFA-Symbols (BOSS) [44]. Another possibility to symbolize the series is via a grid-based representation [92], which places a two-dimensional grid over the time series and generates a bit string describing which values were kept and which bins they were in. A similar idea has been applied in [52, 93] which define a *clipping threshold* to generate a bit over each data point indicating whether the series is above or below the threshold. Clipping-based methods have a tight lower bounding guarantee as the clipped series allows comparing directly with the raw time series.

To summarize, the transformation-based methods apply a set of *rules* or *operations* to transform the *entire* time series into a low-dimensional representation. The parameters in the rules can be constant (i.e., non-data adaptive) or dynamic (i.e., data adaptive) over each time series instance. We should note that each data point in the series contributes to constructing the transformed representation, thus leading to relatively complete conservation of the series' information and a low reconstruction error.

2.1.2.2 Local pattern-based Representations

For most real-world problems, such as time series *classification* or *clustering*, we usually expect the time series samples to be equal in length, with no presence of significant noise and missing values or intervals, etc. The representation built from the entire sequence with such imperfect properties will greatly limit the downstream mining tasks. Therefore, instead of considering the *entire* time series for building the representation, another set of work [94, 95, 96, 97] suggest that the *local* patterns or their combinations can show great power for representing the entire sequence. These representations are usually related with specific mining tasks, such as *classification* or *clustering*.

A new primitive named *Shapelet* was proposed in [14] as a typical local pattern-based representation for time series. Initially designed for *time series classification* task, the *Shapelet* was defined as a subsequence that is maximally representative of a class and can discriminate one class from the others. This representation can be considered as a step forward towards bridging the gap between time series and shape analysis, which has a big advantage on interpretability. A large amount of work extends the Shapelet representations from local-based patterns to the global features of the time series. For instance, [98] applies rule-based modeling on a set of Shapelets and select the best Shapelet combination as the representation; [97] applies Shapelets as individual feature attributes and represents the time series as a set of attribute values containing the distance from the Shapelet to the original series. Instead of extracting Shapelets from the raw time series with high time complexity [14], recent work [99, 100, 101] tend to learn a Shapelet representation from scratch to reduce the complexity while conserving the interpretability. Either the extraction-based [14, 16, 97] or learning-based [101, 99, 102] Shapelet representations (*usually*) show big advantage on the shape-based interpretability. Not limited to the supervised *classification* task, Shapelet can also be applied to the unsupervised *clustering* task. Authors in [94] propose Unsupervised Shapelet (U-Shapelet), which adopts the subsequence (i.e., Shapelet) with appropriate distance with other samples to represent the whole series. A scalable version is proposed in [95] which selects Shapelets based on the SAX representation to accelerate the distance computation. This work is followed by [103] for processing uncertain time series.

Similar to the Shapelet representation, the recurrent local pattern (i.e., *motifs* [37]) is capable of representing a long, repetitive time series. The main difference between the Shapelets and motifs lies in the repetitive characteristics of the data. Authors in [104] detect local patterns in repetitive time-series via fitting local polynomial functions of arbitrary degrees on each sliding window. [96] discovers the motif patterns of variable length and selects the most frequent pattern as the output representation. All these approaches can be combined with the transformation-based representations (e.g., SAX) to accelerate the local-pattern discovery.

2.1.2.3 Model-based Representations

In the model-based representations, we usually assume that the observed time series is generated by an underlying model. The parameters of such a model can be considered as a representation. In this categories, we organize the approaches into two classes: *statistic modeling* and *neural network-based modeling*.

For statistic modeling-based representations, several parametric temporal models have been considered in the literature. For instance, the Auto-Regressive Integrated Moving Average (ARIMA) allows modeling the statistic features (i.e., trend, cycle, stochastic persistence components, and random elements) of a time series with a set of coefficients (e.g., the orders of the seasonal auto-regressive and moving average components, the order of seasonal differencing, etc.). Authors in [105] applies ARIMA model's coefficients as time series representations for *time series clustering* task. Similar techniques such as learning Markov Chain (MCs) representations [106] of the dynamic process in the time series, which is represented as a transition probability matrix, or learning the Hidden Markov Model (HMM) [107] representation for clustering the time series.

The (deep) representations modeled by neural networks have attracted widespread attention recently [17, 108, 109, 30, 29, 110, 111, 112, 113, 11]. A neural network is a graph of computing units (called neurons) structured in layers, which apply a non-linearity on a linear combination of input values [114]. The output of each layer can be considered as a representation that integrates the key characteristics of the input data. The specific domain knowledge (e.g., temporal features, shape-based features) of time series can be used to help design the network for learning representations [27]. Depending on whether the data annotations are used, we categorize the representation methods into two classes: *supervised* and *unsupervised* representation learning.

Supervised neural representations. Most of the neural representations of time series in the literature are learned in a supervised manner. With the help of data annotations, the supervised representations are usually learned for specific tasks, such as classification [115, 116], regression [117], forecasting [4], or even time series retrieval [110]. The main difference between them lies in the final layers of the network and the choice of loss functions for optimizing the model parameters. The first layers of the network can be considered as a feature extractor which can be designed differently with the assumptions or prior knowledge about the time series data. For instance, the convolutional neural networks (CNNs) or recurrent neural networks (RNNs) are two basic modules that can be used to extract time series features from different aspects (e.g., local features from CNNs, temporal features from RNNs). A huge amount of time series networks are built on these two basic units with the inspirations from the computer vision (CV) and natural language processing (NLP) communities. Typical techniques such as adding residual connections (i.e., ResNet [118]) between CNN blocks in [119], adopting the inception structure [120] to combine multi-scale convolution kernels in [121, 122, 123], considering the attention mechanism [124] on CNNs [125, 111] or RNNs [10], etc. The networks can also be designed with

the help of the classical time series analysis. For instance, [126] combines the statistic features extracted from time series with a multi-layer perceptron (MLP) neural network to learn the representation; [127] applies a frequency-filter on the input series to feed the multi-frequency signals to the network; [11] adopts the idea of *wavelet decomposition* to build frequency-aware neural networks; [128] leverages traditional auto-regressive model with CNN-RNN networks to discover long- and short-term temporal patterns.

The supervised networks for learning the time series representation are usually task-specific approaches. In other words, the representations designed for one learning task (e.g., classification) is not always applicable for another one (e.g., forecasting), as they emphasize different characteristics of time series. For instance, the CNN-based models [12, 59] achieve satisfying results on *time series classification* tasks as the local convolutional features can be easily weighted or combined to output a single prediction [3], whereas the *time series forecasting* models [4] are biased towards modeling the long- or short-term temporal patterns to find the correlation between future and historical values, the model structures are generally based on the temporal convolutional networks (TCNs) [129, 130], RNNs [131, 132], or transformers [133, 134].

Unsupervised neural representations. Different from the supervised neural representations, which are task-specific, the unsupervised neural models learn general representations which are ready to be used for a variety of learning tasks.

The first powerful family to learn the unsupervised representations is based on the auto-encoder (AE) [135] structure. The main idea is to train an encoder and a decoder simultaneously, such that the encoded data can be successfully reconstructed. If so, the encoded data integrates the essential characteristics of the input data as its representation. The sequence-to-sequence (seq2seq) auto-encoder [136] is designed for sequential data and has been widely adopted in recent studies [113, 137] for time series representation learning. Various work improved the seq2seq structure in the time series context. For instance, the Deep Temporal Clustering Representation (DTCR) [29] integrates the temporal reconstruction and K-means objective into the seq2seq model to learn cluster-specific temporal representations. This model is further extended by [109] for learning time series with missing values. As shown in [113] by their experiments, if the general representations learned with a seq2seq model are fine-tuned using the downstream classification task, it can significantly improve the performance. Recent studies tend to use only the encoder part in the seq2seq architecture and show that it allows learning more general representations for a multitude of tasks, such as classification, regression, imputation, forecasting, etc. For instance, [111] adopts an CNN-based encoder, [138] adopts a transformer-based encoder. The main focus of these works lies in the design of loss function or the ground truth selection from the data themselves. In addition, using only an encoder allows reducing about half the model parameters, which leads to computational and learning benefits (e.g., avoiding overfitting).

The second category for learning the unsupervised representations is the generative models. The principle is to learn a latent probability distribution that approximates the

data. For instance, based on the auto-encoder structure, the variational auto-encoder (VAE) allows regularizing the representations by introducing a fixed prior on the latent distribution. In this manner, given a distribution (e.g., *Gaussian*), the model is capable of decoding the regularized representation into a generated sample. Authors in [30] learn temporal representations of time series based on a joint learning of a discrete variational auto-encoder (VAE) [139], an self-organizing map (SOM) [140] latent space and a Markov transition model [141]. The learned model shows high interpretability over the two-dimensional representations. Another set of time series generative models [142, 143, 144, 145, 146, 147] are based on the generative adversarial networks (GANs) [148, 149, 150]. GAN learns the data distribution by playing an adversarial game of fooling and discriminating. The generator and discriminator keep improving their performance during the adversarial game. In this way, the GAN-based models are capable of learning and explaining the underlying structure of the input data even without labels. The original study (C-RNN-GAN) [142] applies LSTM on both generator and discriminator for generating music songs. RCGAN [143] adopts Conditional GANs [151] with additional information (e.g., class labels) to guide the sequential data generation process. TimeGAN [144] models the time series on latent representation space, then enhances the temporal dynamics on the generated data based on the concept of Adversarial Auto-encoder [152]. More related studies extend the aforementioned models on time series anomaly detection [145], time series generation [146], or time series imputation [147].

Recently, contrastive learning [153] has become popular for learning time series representations. Being part of the self-supervised learning [154], the contrastive learning adopts the data itself as the supervision, and aims to learn an embedding space in which similar sample pairs stay close to each other while dissimilar ones are far apart. In other words, given an *anchor* time series sample in the learned representation space, the similar (i.e., *positive*) samples should stay close to the anchor; in contrast, the dissimilar (i.e., *negative*) samples need to stay away from the anchor. For unsupervised representation learning on time series, the key questions are how to define the *positive*/*negative* samples and how to build the contrastive training objective. Inspired by the classic word representation learning method known as word2vec [155], authors in [17] propose a novel triplet loss for time series, where the *positive* samples are down-sampled from the anchor samples, the *negative* samples are randomly chosen from the data set. This method is extended by [108] with consideration of non-stationary time series, and by [156] which further minimizes the distance between *positive* (*negative*) samples to accelerate model's convergence and improve model's stability.

To conclude, time series representation learning is a big topic covering almost all the time series mining tasks as shown in Figure 2.1. It has attracted wide attention from the community of database, statistical modeling, machine learning, and so forth to manage and learn the complex time series data. The taxonomy defined in Figure 2.3 provides an overview of the time series representation approaches and their theoretical bases. However, in practice, the representations do not always belong to one single category. For instance,

the SAX-based U-Shapelet [95] is a *transformation-based* and *local pattern-based* representation. The adversarial Shapelet [101, 157] is a *local pattern-based* and *model-based* representation. A hybrid representation applies to the complex context where multiple requirements are raised, such as computational/storage cost, interpretability, etc.

2.1.3 Data Stream and Time Series

When we talk about time series mining, we generally consider the mining activities on a static time series database. In practice, the time series are usually collected in a dynamic context, where the data comes continuously with the characteristics of the *data stream*.

Definition 2.15. (Data stream). A *data stream* S is a continuous input data where each instance can by any type of data, such as row data, image, text, etc.: S=(x_1, x_2, ..., x_N), where N increases along with the time.

One important notion in the data stream is *concept*, which refers to the data distribution of the streaming instance. It can be the distribution $p(x)$, conditional distribution $p(y|x)$, or joint distribution $p(x, y)$, where x is the streaming instance, y is the related label (if available). We take the *Time Series Classification* (TSC) as an example. In TSC tasks, we intend to predict the label of a new input TS instance by extracting the knowledge from the collected data. The optimization of TS feature extraction and model construction process allows us to strive for a low prediction error, and approach to Time Series' nature Concept, i.e., the conditional distribution $p(y|x)$. When the research context extends to the data streams, the knowledge base is no longer constant and evolves gradually with new input data. The challenges here can be represented by the following five intrinsic characteristics of Data Stream [158, 159, 160]:

- **Infinite Length**: The continuous streaming data is infinite, thus more requirements for analyzing the data are raised, such as computational efficiency, model/data storage cost, etc.
- **Feature Evolution**: The feature space changes over time, new features become useful and old ones may become redundant. The learning model is intended to extract the latest features to approach the inner concept of the data source.
- **Concept Drift**: The prediction target or the conditional data distribution $p(y|x)$ evolves over time, the learning model must be capable of adjusting itself gradually to the most recent data by an effective process of Concept Drift detection.
- **Label Constraint**: In practice, it is usually hard, even impossible, to annotate all the streaming instances in real-time. Therefore, part of the data may be unlabeled.
- **Concept Evolution**: Non-labeled instances with novel classes may emerge in the future, which should be recognized and separated from existing classes.

In the literature, many works consider the streaming characteristics for time series analysis, which covers a huge amount of real-life applications. However, due to the vast time series mining activities as shown in Figure 2.1 and the complex characteristics of the data stream, it is hard to have a clear idea about how the analysis on the intersection of these two domains (i.e., time series and data stream) works, and in which contexts they are applicable [161]. Therefore, in this section, we divide the related work into two categories according to the data format, for which we give two formal definitions: *time series stream* and *streaming time series*.

2.1.3.1 Time series stream

Definition 2.16. (Time series stream). A *time series stream* S_{TS} is a continuous input data stream where each instance is a Time Series: S_{TS}=(T_1, T_2, ..., T_N), where N increases along with the time.

For Time series stream, the information should be extracted from each new input TS instance and be merged with the existing learning model. By considering various challenges in Data Streams, we are capable of launching the analysis from different contexts: 1) Within a stationary concept, the learning model tries to make the learned concept stay as close as the real one. Feature Evolution concerns the incrementality of the learning algorithm, which is the necessary condition for the stream learning system; 2) Further consideration of the non-stationary concepts, like Concept Evolution and Concept Drift, which can be respectively adopted in more dynamic contexts where data instances are weakly labeled, or prediction target evolves over time.

[Application Scenarios] In typical TS learning tasks, such as TS classification [115], the analysis process is launched from an off-line dataset, without considering the context of a dynamic data source. Dynamic or streaming data sources usually result from monitoring applications, but require online training, that said *Training online*, *Monitoring online*. For instance, domains like health care look to enrich the database gradually with more medical cases, i.e., Feature Evolution; In astronomy, with human's growing knowledge about the universe, the theoretical basis for labeling the light curves (TS) will change, i.e., Concept Drift, meanwhile, the light curves collected from unknown supernovas may be imported into the dataset, i.e., Concept Evolution. Most Time Series data, such as sensor readings, are labeled during the data collection process. The post-labeling on TS is much more costly than classic data (e.g., image, text, etc.) due to the low interpretability over the real-valued sequence, i.e., Label Constraint. The techniques applied in an off-line fully labeled TS dataset are then not adaptable in such dynamic scenarios, that said streaming context.

Most of the work [162, 163, 164, 9] related to time series stream in the literature considers the *classification* problem. For instance, authors in [162] apply Nearest Neighbor (NN) algorithm for classifying time series stream with humans in the loop. The costly NN

model can be interrupted anytime, which would be more accurate with more time being consumed. This work was extended in [164] which buffers a set of streaming instances as the learned model. A scoring function is used to estimate the intermediate result quality of an incoming instance. In this manner, the model buffer can be updated accordingly. To improve the model efficiency, Kasetti et al. [163] represents the time series as bitmaps, [9] transforms the time series instance to a SAX dictionary [91].

2.1.3.2 Streaming time series

Definition 2.17. (Streaming time series). A *streaming time series* S is a continuous input data stream where each instance is a real-valued data: $S=(t_1, t_2, ..., t_N)$, where N increases with the time.

We consider Time Series as a **local collection** of real-valued data from an online streaming source generating a never-ending data flow, then *Streaming Time Series* represents such an online streaming source. Different from *Time Series Stream*, mining from *Streaming Time Series* has distinct objectives: 1) A learning model is usually built upon an off-line labeled data set to **monitor** the data flow coming in real-time, that said *Training off-line*, *Monitoring (or inference) online*; 2) The model is **learned** from the data flow coming in real-time, which is followed by immediate inference. We call it *Training online, Monitoring (or inference) online*.

[Application Scenarios] For the *Training off-line*, *Monitoring online* mode, the applications like patient's ECG monitoring, a known ECG (TS) feature set is learned or known in advance, which allows the system to focus on the real-time processing [165] on the input data; the activity recognition [166] is another typical application, where the pre-known activity patterns (i.e. *motif*) serve to recognize in real-time the activities over Streaming TS. In these scenarios, we usually focus on how to improve the model efficiency for processing the fast-coming data stream. For the *Training online*, *Monitoring online* mode, the streaming time series data can be either labeled (*supervised*) or unlabeled (*unsupervised*). First, for *supervised* tasks, since considering the individual real-valued data point (or instance) as a supervised object is meaningless, the annotations are usually given to the streaming segments. **This leaves the research problem being equivalent to the supervised learning on the time series stream**; Then, for unsupervised tasks, one typical scenario is the *online motif discovery*, where the frequency of the repeated patterns (i.e., *subsequence*) evolves along time, i.e., Feature Evolution. One particular mining activity on streaming time series is *online forecasting*, which can be considered either as *supervised* (the annotations, i.e., future values are required), or *unsupervised* (no external annotations are needed). One way to train such an online model is to regularly cache the most recent samples as the annotations for past data.

Most of the work [167, 165, 168] in the literature studied the *Training off-line*, *Monitoring online* mode for streaming time series. For instance, [167] queries a subsequence

in the streaming time series by designing an optimal distance measure; Similarly, [165] transforms the data with PAA representation [45] to improve the query's efficiency; [168] parallelizes the querying process with the popular distributed frameworks (Kafka, Spark Streaming). As for the the *Training online*, *Monitoring online* mode, the work usually studies the *online motif discovery* task, where the TS patterns (or *motifs*) can be detected and updated on the fly [169]. For instance, [170] just find the motifs based on the most recent data in streaming time series; instead of considering the frequent motifs, [40] only find any *pair* of repeated patterns (i.e., *rare motifs*). One particular work in [171] considers both *unsupervised motif discovery* task and *supervised classification* task on streaming time series. The authors designed an *active learning* [172] model with humans in the loop. The system firstly discovers the frequent motif and asks for human for a label. The labeled motif is considered as a *local pattern-based representation* of the data, which is added into the model and serves to monitor the newly coming data. In this way, the *unsupervised motif discovery* and *supervised classification* is linked up with the human in the loop.

2.1.4 Semi-supervised Learning on Time series

In this section, we present the basic concepts related to semi-supervised learning (SSL). Specifically, we review the work of semi-supervised learning on time series data.

Definition 2.18. (Semi-supervised learning). The semi-supervised learning (SSL) aims at training a learner f (e.g., classifier, regressor, etc.) to predict successfully the label of a testing sample, adopting the supervised training from the labeled samples and further unsupervised adjustment/optimization from unlabelled samples.

The main advantage of the semi-supervised learning is that it requires less human effort for the costly data labeling and generally achieves higher accuracy than only using the labeled data. There are many semi-supervised learning methods in the literature, which can be organized into six categories: SSL with generative models, SSL with low-density separation, graph-based methods, co-training methods, self-supervised methods, and self-training methods [173]. We specifically review works that have related to the semi-supervised time series classification.

The complex time series data, such as sensor readings, are labeled during the data collection process. The post-labeling of time series is much more costly than classic data (e.g., image, text, etc.) due to the low interpretability over the real-valued sequence. As a result, there have been extensive efforts to apply semi-supervised learning algorithms explicitly designed for time series classification [173, 174, 175, 176, 177, 178].

The pioneering work [173, 174] on Semi-supervised TS Learning are based on self-training or Positive Unlabeled Learning [175] with the Nearest-Neighbor (1NN) classifier and a carefully designed distance, such as DTW [173] or DTW-D [174], and optimized stopping criterion [176] for importing the pseudo-labels. Though not mentioned in their papers, the self-training framework is extensible from the univariate time series (UTS) to

the multivariate time series (MTS) by using an adapted distance measure, such as DTW_I [179], DTW_D [179] or DTW_A [180]. However, under more complex scenarios nowadays, such as 128 UCR UTS datasets [181] and 30 UEA MTS datasets [182] collected from different domains, the distance-based classifiers show limited performance and are rather used as baselines by recent studies [115, 116].

Beyond the self-training methods, the authors of [183] proposed a graph theoretic SSL algorithm that constructs graphs relating all samples based on different distance measures such as DTW or Wavelet Transform [67], and consequently propagates labels from the neighbors. Another work [177] applies shapelet learning [99] on both labeled and unlabeled time series data. The learned shapelets are used to classify the unlabeled samples, thereby producing pseudo-labels. A coordinate descent solver wraps the optimization process by iteratively solving for the classification of labeled samples, pseudo-labels, and shapelet learning, respectively. Authors in [184] adopt the concept of *Transfer Learning* [185] and apply the domain adaptation on the weakly labeled data. Specifically, given the labeled data in *source* domain and the unlabeled data in *target* domain, the knowledge that we learned from the source domain can be transferred to the target domain with an unsupervised domain adaptation. This is especially practical for complex time series datasets having high variability between domains. By adopting the concepts of Multi-Task Learning (MTL) [186] and self-supervised learning [154], authors in [178] learns the self-supervised UTS features from an auxiliary forecasting task, which help improve the classification performance of time series. The recent work Semi-TapNet [187] proposes an Attentional Prototype Network to learn from the unlabeled samples, for which the pseudo-labels are predicted by intermediate-trained classifiers.

Recent studies tend to learn meaningful time series representations [27] in a weakly supervised setting and have drawn much attention in the domain. Unsupervised Scalable Representation Learning (USRL) described in [17] combines causal dilated convolutions [129] with triplet loss for contrastive learning [153]. On the one hand, it learns a better representation of univariate time series than the traditional supervised CNN model [119]. On the other hand, a single SVM on the learned multivariate time series representation offers higher accuracy than a DTW_D-based classifier [179].

2.2 Time Series Representation for Classification

In this section, we review the work which builds or learns time series representation for a concrete classification task. As mentioned in Section 2.1.2, the time series representation can be in single form of *transformation-based*, *local pattern-based* or *model-based*, or a combination of them for different research purposes in a complex context.

It the literature of the *time series classification* (TSC) [115, 116], various TSC approaches have been proposed by researchers in recent years which are suitable for different contexts along with *dissimilar TS feature representations*. The time series representations

can be generally divided into five categories:

- Raw time series representation: including the global feature of entire series [188] for One Nearest Neighbor (*1-NN*) classifier, which is usually combined with various distance measures [189, 190, 188, 191].
- Statistic summary representation: the raw time series is transformed into the summary statistic features (e.g., mean, deviation, slope, etc.) extracted from every sub-series or the entire sequence. Various classifiers [192, 193, 194] can be built on top of the statistic features.
- Local pattern representation: the local pattern in the time series can contain the key characteristics of the entire sequence; thus, the raw time series can be represented by one or a combination of local patterns.
- Deep representations: the last layer of a neural network classifier [3] embeds the TS features, various neural network structures are designed with the inspirations from the classic time series analysis and recent advances of computer vision (CV) and natural language processing (NLP) communities.
- Ensemble representations: various time series representations (e.g., global feature-based, frequent motif-based, Shapelet-based, etc.) can be assembled to characterize the time series from different aspects, i.e., ensemble approaches [188, 7, 195].

In the following sections, we review the previous *time series classification* work in each of those categories.

2.2.1 Raw Sequence as Representations

When we adopt the raw sequence as the TS feature representation, the model considers the global sequence feature based on various distances to measure the similarity between different time series. Once the distances are defined, they can be combined with multiple machine learning models such as *Nearest neighbor* classifiers for specific tasks. We present here several classic distance measures for time series data.

2.2.1.1 Euclidean distance

The Euclidean distance (ED) is the most commonly used distance to measure the similarity between two time series. Given two time series x, x' of the same length n, their *Euclidean Distance* is defined as follows:

$$ED_{x,x'} = \sqrt{\sum_{i=1}^{n}(x_i - x'_i)^2} \tag{2.1}$$

The Euclidean distance with the nearest neighbor classifier is always adopted as the baseline of the TSC tasks. It considers the time series as a general data vector and provides an exact mapping between each time point in two time series with time complexity $\mathcal{O}(n)$. Consequently, the temporal information is essentially ignored. In general, this distance requires the time series to be equal-length when applying the nearest neighbor classifier. For time series with different lengths, it can be done by measuring the distance between the closest subsequence pairs, which introduces extra computational cost with a sliding window technique.

2.2.1.2 Dynamic time warping (DTW)

The Euclidean distance does not consider the temporal information in time series. Precisely, the temporal scaling or the local distortions of the time series make the Euclidean distance inapplicable in most real-life applications. Dynamic time warping (DTW) [196] is an elastic similarity measure for the sequential data, which can provide an elegant solution to tackle those problems. As shown in Figure 2.4, the DTW measure allows warping of the time axis by finding an optimal alignment between two time series. Specifically, a distance matrix is defined where DTW aims to find the shortest warping path as the optimal alignment. A warping path W is a set of contiguous matrix indices defining a mapping between the time points in two time series. In the distance matrix, there are enormous possible warping paths, and the optimal path is the one that minimizes the global warping cost/distance. DTW can be computed using dynamic programming [196] with quadratic time complexity $\mathcal{O}(n^2)$, where n is the length of time series.

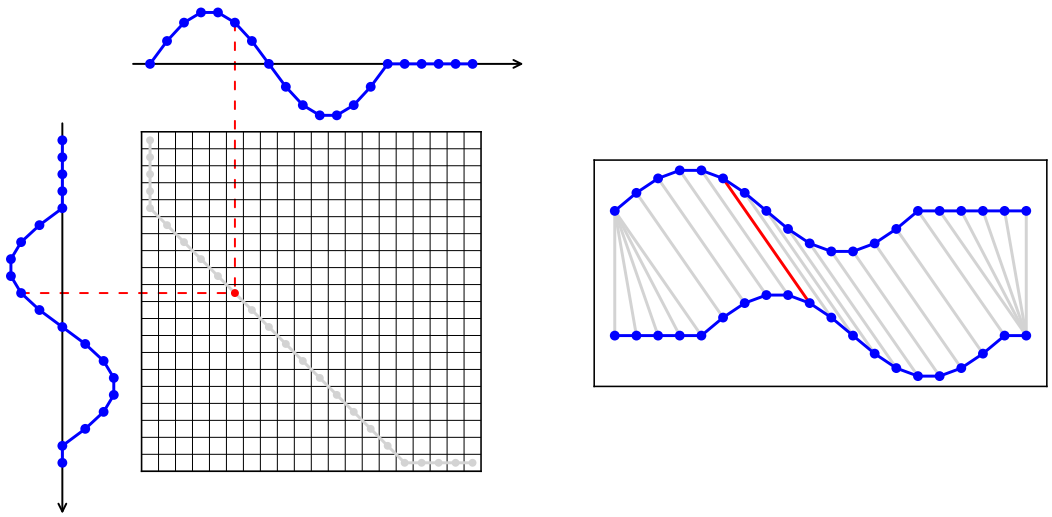


Figure 2.4: DTW example for two series X and Y: optimal warping path (left) and alignment (right).

Since DTW provides an elastic mapping between sequence values, it can be applied

to align two time series with different lengths. Due to its quadratic complexity, DTW is considered as a costly distance measure compared to the Euclidean distance. A large set of works have improved DTW for better efficiency performance, which are generally based on lower bounding measures. In other words, it is possible to put a restriction on the amount of the allowed warping. A maximum allowable distance between any pairs of indexes is defined to lower bound the warping path. For instance, authors in [197] propose the notion of the *upper and lower envelope* that represents the maximum allowed warping in the distance matrix. The complexity then becomes $\mathcal{O}(n)$. The FastDTW proposed in [198] calculates the warp path in a multi-resolution manner, which recursively projects a warp path to a higher resolution and then refines it. The multi-resolution approach allows a linear-time computation of DTW but with an exchange of information loss leading to a second optimal warping path. Many other DTW variants improve the DTW measure to tackle various practical problems, such as noisy time series. For instance, [199] proposes Derivative Dynamic Time Warping (DDTW) that first transforms the series into a series of first differences. Precisely, given a series $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, the difference series is $\mathbf{x}' = \{x'_2, x'_3, \ldots, x'_{n-1}\}$ where x'_i is defined as the average of the slopes between x_{i-1} and x_i with another average value between x_{i+1} and x_{i-1}, i.e. $x'_i = \frac{(x_i - x_{i-1}) + (x_{i+1} - x_{i-1})/2}{2}$.

2.2.2 Statistic Features as Representations

Being part of *non-data adaptive* representations, the statistic features can be extracted from the time series at the sequence level or the interval level. For the statistic representations in the sequence level, the summary features are extracted from the entire series. For instance, authors in [126] started to discuss the extraction of basic features such as max, min, skewness over the whole series, which can be combined with various classifiers such as the Support Vector Machine (SVM) or multi-layer perceptron (MLP) neural network to conduct the predictions; Similarly, [200] extracts the wavelet features as the global feature representations to feed a neural network classifier; [201] extract the general patterns such as the peak values to represent the whole series, which can be combined with the decision trees to do interpretable predictions. An extensive work [202] even collects more than 9000 features generated from various algorithms that have been discussed in fields such as medicine, astrophysics, finance, mathematics, climate science, industrial applications, and so on. A Python library named FATS is proposed in [203] which facilitates and standardizes feature extraction for time series data. Even though FATS initially focuses on the feature extraction for astronomical light curve data, it is generalizable for other time series data. The statistic feature representations are always applied with a feature selection process and can be combined with various classifiers. For instance, a greedy forward feature selection [204] can be adopted, which grows a set of important features incrementally by optimizing the classification performance on the training data.

The statistic representations can be built as well at the interval level. Similar features, as mentioned previously, can be extracted from each time series interval. For instance,

the time series forest (TSF) [192] used measures of mean, standard deviation, and slope in local equal-length intervals of time series to build a set of decision trees within the random forest architecture, where a decision tree is built on a random selection of interval features. Similar to TSF, the Time Series Bag of Features (TSBF) [193] extracts the statistical features from the intervals with different lengths. The interval features are considered as the new attributes of the original time series.

2.2.3 Local patterns as Representations

As mentioned in Section 2.1.2.2, the local time series patterns can be adopted to represent the whole series. For the time series classification (TSC) tasks, two types of patterns are studied in the literature: *Shapelet-based* and *motif-based*.

2.2.3.1 Shapelet-based representation for TSC

Shapelet [14] is the most commonly used local pattern for classifying time series. The shape-based representation is defined as a sub-series that is maximally representative of a class and can discriminate one class from the others. As shown in Figure 2.5, the shapes of the projectile points can be converted to the time series using the angle-based method [205]. The initial Shapelet work [14] builds a Shapelet dictionary composed of the Shapelets of different classes with their optimal split point values. Figure 2.5 show two Shapelets extracted respectively from the *Clovis* and *Avonlea* classes. A binary decision tree classifier can then be built on the class-specific Shapelets. For instance, given a time series $\mathbf{x}$, if its distance with the shapelet-*Clovis* is smaller than 11.24, $\mathbf{x}$ belongs to the *Clovis* class; otherwise, if its distance with the shapelet-*Avonlea* is smaller than 85.47, $\mathbf{x}$ belongs to the *Avonlea* class; otherwise, the time series $\mathbf{x}$ belongs to an unknown class.

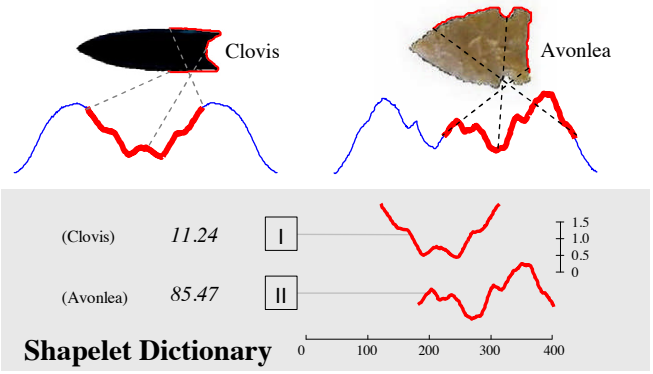


Figure 2.5: Shapelets (in red) from [14]. The projectile points can be converted into the time series, where the class-specific Shapelets are *extracted* with their optimal splitting points (i.e., 11.24 for *Clovis* and 85.47 for *Avonlea*). A binary decision tree can be built on the Shapelets and the splitting points to classify the time series.

The Shapelets extracted from the raw time series usually show great interpretability over the shape-based representations but with high time complexity $\mathcal{O}(N^2T^4$, where N is the number of training instances, and T is the length of the TS instance. Various work has improved the extraction-based Shapelets either on the efficacy or on the efficiency. For the efficacy aspect, [98, 97, 206] combine a set of Shapelets as the representation, [207, 208] propose the advanced quality measures for evaluating and selecting the Shapelets; as for accelerating the Shapelet extraction process, the (local pattern-based) Shapelet representation is always combined with the *transformation-based* representation on time series. For instance, Fast Shapelet (FS) [209] adopts SAX [86] and random projections [210] to accelerate the Shapelets' extraction. The *transformation-based* representation helps reduce the data dimensionality of time series, thus leading to a better efficiency performance. However, these efficiency improvements are usually obtained at the cost of a lower classification accuracy [115].

In order to tackle the high time complexity issue of the extraction-based Shapelet, the learning-based Shapelet is studied in recent work [99, 100, 101, 157]. The landmark work [99] shows that the Shapelet can be learned from scratch with a gradient-descent-based optimization algorithm. Figure 2.6 shows the model's optimization process when learning the Shapelets. The Shapelets can be considered as a set of parameters to be learned. The idea is to jointly learn the optimal shapelets and the optimal linear hyper-plane W (i.e., a logistic regression classifier) that minimizes the classification objective.

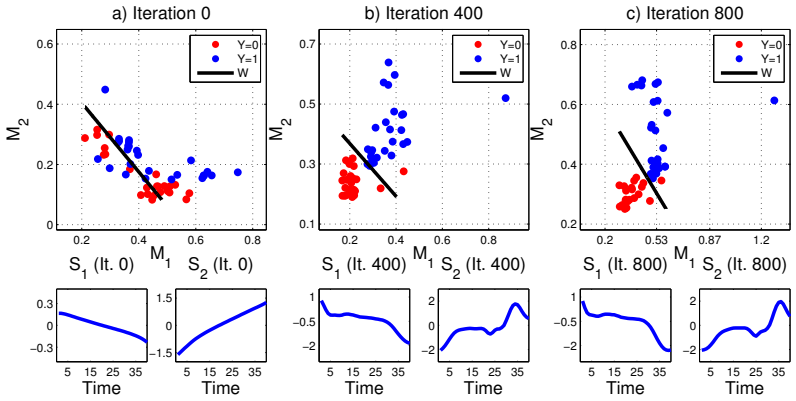


Figure 2.6: (Figure taken from [99]) Two Shapelets *learned* from the Gun-Point dataset [181]. The Shapelets can be progressively learned at each training iteration to reduce the loss error of a logistic regression classifier.

Learning Shapelet (LS) has a time complexity of $O(N^2m^3)$, where N is the number of TS instances in the training set, m is the length of the Shapelet to be learned. LS shows higher accuracy and lower time complexity compared to the extraction-based Shapelets. However, the learned Shapelets are not always similar to the sub-series in the original TS, thus losing the interpretability. Therefore, various work [101, 157, 211] studied how to learn interpretable Shapelets from scratch. For instance, the Robust Learning Time-series

Shapelets (RLTS) [211] modifies the feature vector of LS by adding Robust Losses to model the influence of unreliable instances, a shapelet regularizer is proposed for approximating the learned shapelets to the best-matching segments of the reliable TS instances. Authors in [101, 157] adopt similar adversarial training techniques to filter the learned Shapelets which are dissimilar to the segments in raw TS, thus guiding the Shapelet learning process.

2.2.3.2 Motif-based representation for TSC

When the time series represents a repetitive pattern, it is called *repetitive time series*. The recurrent local pattern (i.e., motif) is capable of representing such a long and repetitive time series. For a real-world time series, it may be fully repetitive or partly repetitive. Depending on the repetitive features in time series, the motif-based representation can be divided into two classes: (i) top-k frequent motifs as representations; (ii) dictionary representations with a set of motifs and their frequencies.

Authors in [96] discover the motif patterns of variable length. The method, referred to as Representative Pattern Mining (RPM), combines with SAX representation to accelerate the finding of the repetitive patterns and considers the most frequent (i.e., *top-k)* motifs as the time series representation. Another set of work [104, 9] constructs a dictionary of the motifs to capture the global frequency distribution over various motifs. For instance, authors in [104] detect local repetitive patterns in time-series via fitting local polynomial functions of arbitrary degrees on each sliding window. The coefficients of the polynomial functions are converted into symbolic forms (i.e., alphabet words). A histogram of the frequencies of the words is constructed from each time series. The nearest neighbor classifier is applied with the Euclidean distance, which computes the difference between the histograms. Similarly, [9] firstly transforms the time series to a SAX representation, then constructs a SAX-word histogram with the related frequency.

2.2.4 Deep Representations

As a powerful feature extractor, the neural networks have been validated in classification tasks for various data, such as image [212], text [213], or medical data [214]. The neural networks have been greatly studied for time series classification tasks. The classic time series analysis [26] can be considered as a guideline to design the network structure. For instance, temporal features such as the trend, seasonality are critical for modeling the financial time series. The classic techniques such as the time series decomposition [215] can be adopted, which decomposes the time series into trend components and seasonal components. The neural networks can be designed accordingly using the recurrent structure (e.g., RNNs) to extract the temporal patterns. When time series is collected from the context (e.g., medical data) where the frequency features are representative, the frequency-aware networks [11] can be designed accordingly.

Enormous network architectures have been recently developed for time series classifi-

cation (TSC). The convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are two basic modules that are combined with various techniques to build the network architectures. These techniques are inspired either by the classic time series analysis [26] to explore the time series' characteristics from different aspects, or by advanced network structures which have been validated in other domains, such as computer vision (CV), natural language processing (NLP). However, the most advanced TSC networks [59, 121] usually consider a set of techniques to design a generalizable model to various application domains, such as the 128 UTS datasets in UCR archive [181].

The classical time series analysis provides great inspiration for designing the TSC networks. For instance, [126] combines the statistic features (e.g., max, min, skewness) extracted from time series with a multi-layer perceptron (MLP) neural network to learn the representation; [127] firstly decomposes the time series into multi-frequency signals with a frequency filter, then applies multi-channel network to extract and combine the features from each of the signals; Similarly, the spectral analysis has been applied in [11], which adopts the idea of *wavelet decomposition* to build frequency-aware neural networks; the recent CNN-based models (e.g., ROCKET [12, 59] apply thousands of random convolutional kernels to explore thoroughly the local features in time series.

The advanced network architecture from other domains (e.g., CV, NLP) can help build a time series representation for a better prediction performance. For instance, authors in [119] apply the residual connections (i.e., ResNet [118]) between CNN blocks to eliminate the gradient vanishing issue when training the network; [121, 122, 123] adopt the inception structure [120] to combine multi-scale convolution kernels to explore the local features from different scales; Furthermore, the attention mechanism [124] can be adopted to optimize the network structure. Based on the attention mechanism, [125, 111] consider weighting the local time series features extracted by CNNs, authors in [10] weight the temporal features extracted by RNNs in each time point.

2.2.5 Ensemble Representations

The ensemble representation is a set of time series representations that are learned from various aspects. An ensemble of classifiers is built on the individual time series representation to output a single prediction. In contrast, the hybrid representation that we mentioned previously builds a single representation on top of various TS representations. For instance, Fast Shapelet (FS) [209] adopts SAX [86] (*transformation-based* representation) and Shapelet [14] (*local pattern-based representation*) to learn a single hybrid representation. A single classifier is applied over the hybrid representation.

The ensemble representation combined with various ensemble models usually gets better generalization performance than individual representations. For instance, the elastic ensemble (EE) [188] combines 11 one-nearest-neighbor (1NN) classifiers with various similarity measures, such as Euclidean distance, Dynamic time warping (DTW) [196] or its variants. Essentially, EE is based on the raw TS representation with various classifiers.

Similarly, the temporal dictionary ensemble (TDE)[216] combines various dictionary-based TS classifiers (BOSS [44], cBOSS [217], S-BOSS [218], WEASEL [219]), for which the representations are either Symbolic representation (e.g., SFA [90]) or Spectral decomposition-based representation (e.g., DFT [62]). The collection of transformation ensembles (COTE) [6] adopts 35 base classifiers to have a single prediction on several TS representations: raw representation, DFT [62], and Shapelet [14]. The hierarchical Vote Collective of Transformation-Based Ensembles (HIVE-COTE) [7] is built on top of several ensemble models leading to a hierarchical structure. The recent HIVE-COTE versions [220, 221] replace the outdated ensemble modules with the state-of-the-art ensemble methods, to reduce the computational complexity and increase the prediction accuracy.

For ensemble representations, the classification models are biased towards the selection of the TS representations and the techniques for assembling the representations. The classifier's performance strongly depends on the individual classifiers and the complementary between the selected time series representations.

2.2.6 Univariate *versus* Multivariate Time Series

From Univariate Time Series (UTS) to Multivariate Time Series (MTS), the key research question is how to capture the interactions between the variables. Traditional methods usually combine the compact and effective features from different variables; in other words, the representation approaches developed earlier on UTS [31] can be further extended to the MTS context. For instance, authors in [222] explored Singular Value Decomposition (SVD) with multi-view learning to find the consistency and interactions between variables. Similarly, [223, 102] combine Shapelet representation from different variables to build an ensemble-like learner. Symbolic Representation for Multivariate Time Series (SMTS) [224] adopts the Bag-of-Patterns concept, considering all variables simultaneously, and constructs a code-book to model the variable relations. Finally, WEASEL+MUSE [225] extend WEASEL [219] from UTS to MTS by creating a histogram of feature counts to capture the local and global changes in relationships between variables.

The deep learning models [121, 12] designed for UTS classification can be naturally adapted for MTS classification. Specifically, the input channel size of the network should be adjusted to the variable numbers of the MTS. However, such models do not explicitly consider the variable relationships in MTS. In other words, the variables are equally considered for building a classifier. Recent neural network models start modeling the variable interactions with different techniques. For instance, the Multi-Channels Deep Convolutional Neural Networks (MC-DCNN) [226] extract firstly 1D-CNN features from each variable, then combine them with a Fully Connected (FC) Layer. Whereas the authors in [227] abandon the combination strategy but apply directly 1D-CNN to all variables. The 2D-CNN features with the cross-attention mechanism in CA-SFCN [125] enhanced the dependencies captured by 1D-CNN on both temporal and spatial axes. Besides, the recurrent models are widely applied to sequential data. A modified Gated Recurrent Unit (GRU) described

in [21] models MTS with missing values, where each multivariate step is memorized into state units, then the recurrent structure captures the temporal dependencies. Another group of works [228, 229] adopt Graph Neural Networks (GNNs) to model the spatial interactions, which usually rely on external information (e.g., the road networks) between the variables. Last but not least, the hybrid LSTM-CNN structure is capable of extracting both local and long-term features. Various work such as the Squeeze-and-Excitation block in MLSTM-FCN [230] or the multi-view learning-like module in TapNet [187], enhanced the hybrid structure via modeling the variable interactions with a multi-view learning-like structure [222].

2.3 Geo-located Time Series Representation for Forecasting

Different from the time series classification task, time series forecasting tends to find the correlation between future values and past observations. As a classical research problem, the forecasting task has been studied for decades by researchers from various domains, e.g., statistic modeling, machine learning, etc. The proposed forecasting models have been widely deployed to countless real-life applications [231], such as energy consumption prediction [232], financial analysis [233], sales forecasting [234], weather forecasting [235], air quality prediction [236], traffic forecasting [237], etc. From the classic statistic models to recent deep learning models, a huge amount of studies have been conducted for data from various domains. Readers are invited to check the recent surveys of the classic statistic forecasting models in [238] and the deep learning models in [239, 4] to have a general understanding of the techniques for time series forecasting. In a nutshell, the *model-based* representations as shown in Figure 2.3 are usually adopted for time series forecasting tasks.

Recently, several open-source frameworks have been built which help users to quickly get started with existing forecasting algorithms. For instance, *pmdarima* [240] is a Python & Cython wrapper of several different statistical and machine learning libraries (statsmodels [241] and scikit-learn [242]), and operates by generalizing all ARIMA (AutoRegressive Integrated Moving Average) models [238] into a single class; *PyFlux* [243] is a forecasting framework integrating various probabilistic models, such as Vector Auto-Regressions (VARs) [244], Gaussian state space models [245], Generalized Autoregressive Score (GAS) models [246, 247], etc.; The recent proposed *Darts* [248] is a Python library for easy manipulation and forecasting of time series. It contains a variety of models, from classics such as ARIMA [238], Prophet [249] to deep neural networks, such as TCNs (Temporal Convolutional Networks) [129], TFTs (Temporal Fusion Transformers) [133].

Theses frameworks integrates the models for general forecasting problems on time series. However, the forecasting task on a concrete context requires considering certain specific characteristics of data, which help improve the model's performance. For instance, the forecasting models can be designed for: (i) *short-term* or *long-term forecasts*; (ii) *uni-*

variate or *multivariate* time series; (iii) *stationary* or *non-stationary* time series; (iv) time series *with* or *without* missing values; (v) time series *with* or *without* external information; and so on. Therefore, the problem definition is always the most difficult but important part of forecasting [238]. Defining the problem carefully requires an understanding of the collected data and the way the forecasts will be used, who requires the forecasts, and in which context the forecasting will be applied.

In this section, we fix our attention on the *Smart City* context where the time series is collected from the geo-located sensors. For which the research problem is formulated as **geo-located time series forecasting**. We first give the relevant definitions of the forecasting problem. Then, we present the related work in the literature for geo-located time series forecasting.

2.3.1 Definitions

Definition 2.19. (Geo-located time series). A geo-located time series (GTS) $\mathcal{X}=\{\mathbf{X}_t\}_{t=1}^{T} \in \mathcal{R}^{N\times F\times T}$ is essentially a multivariate time series tensor with data collected from a spatial network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, ..., v_N\}$ is a set of N spatial nodes and $\mathcal{E} = \{e_1, ..., e_E\}$ is a set of E edges connecting the nodes. Each spatial node includes T time points each containing F features.

The main difference between the geo-located time series (GTS) and multivariate time series (MTS) is that GTS considers the spatial locations of the time series. The univariate time series in GTS can be collected from the same or different spatial location (node). The spatial node network will introduce external information or characteristics for defining the correlations between the variables. Therefore, the representation learning on GTS requires not only considering the essential features of MTS but also modeling the spatial correlations introduced by the spatial network. A typical application scenario is the *traffic forecasting* [250], where the sensor nodes capture the traffic features (e.g., traffic flow, speed, occupancy) from different locations. At a certain time point, the traffic features in one node are hugely impacted by that in other nodes. The spatial distance and node connections are critical factors affecting node interactions.

Definition 2.20. (Geo-located time series forecasting). Given a training set $\mathcal{D} = \{\mathcal{X}_{1:T}^{(i)}, y^{(i)}\}_{i=1}^{K}$ with K geo-located time series $\mathcal{X}_{1:T}^{(i)} = (\mathbf{X}_1^{(i)}, \mathbf{X}_2^{(i)}, ..., \mathbf{X}_t^{(i)}, ..., \mathbf{X}_T^{(i)})$, their associated label (target variable) $y^{(i)} = (\mathbf{X}_{T+1}^{(i)}, ..., \mathbf{X}_{T+\tau}^{(i)})$, and a spatial network $\mathcal{G}$, we aim to learn a forecasting model f such that $f(\mathcal{X}^{(i)}) \approx y^{(i)}$, in order to forecast precisely the future values of a new testing geo-located time series. The time point $T + 1$ is referred to as *forecast start time* and $\tau \in \mathcal{N}_{>0}$ is the *forecast horizon*. When $\tau = 1$, we call it *single-step forecasting*, otherwise we call it *multi-step forecasting*.

The geo-located time series is a kind of Spatio-temporal data that records temporal information on fixed spatial locations. In the literature, the *geo-located time series*

forecasting and *Spatio-temporal forecasting* are usually used interchangeably, even though the latter covers many more contexts with various Spatio-temporal data. In general, the *Spatio-temporal forecasting* can refer to (i) the geo-located time series forecasting only with time series data on fixed locations [237]; (ii) the geo-located time series forecasting with rich contextual features, such as traffic flow forecasting with external factors of weather conditions [251], public holidays [252], or POIs (Point-of-Interests) features [253]; (iii) the trajectory location prediction [254, 255], where the data has dynamic spatial locations at different time stamps; (iv) the image series forecasting, it can be the satellite image [256], and radar wave photo [257] which adopt the geographical locations as spatial information, or the video images [258] which consider the objects' relative positions as spatial information; and so on. In the next section, we briefly present the literature of the Spatio-temporal forecasting on geo-located time series, where the time series data are collected from fixed spatial nodes. If not specifically mentioned, the *Spatio-temporal forecasting* in this thesis indicates the *geo-located time series forecasting*.

2.3.2 Geo-located time series forecasting

Accurate geo-located time series forecasting has played a critical role in various information systems, such as intelligent transportation, retailing goods distribution, and public risk prevention. For example, people flow prediction can help the supply chain and retailers to schedule the delivery and manage the stocks; traffic volume prediction can help the transportation department better manage and control the traffic to ease traffic congestion.

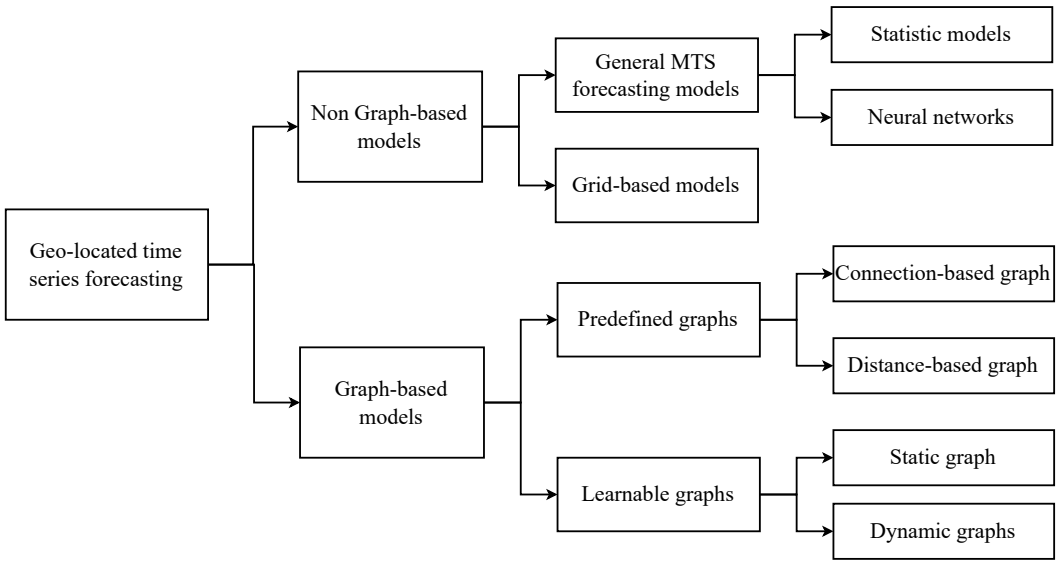


Figure 2.7: Taxonomy of geo-located time series forecasting approaches

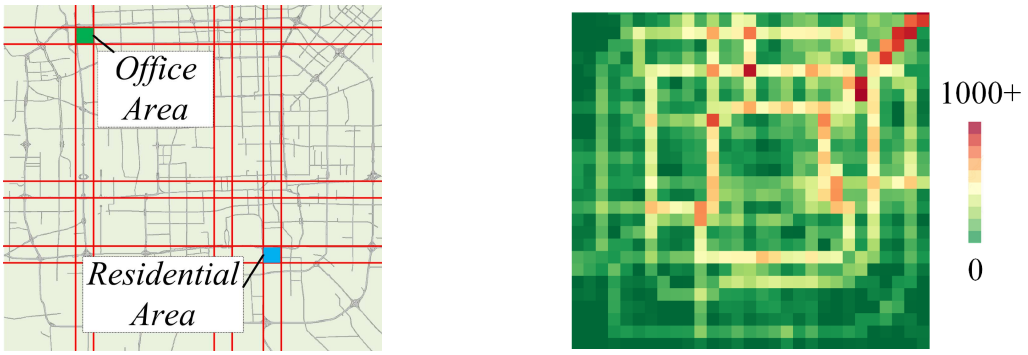
Different from many multivariate time-series (MTS) forecasting problems, geo-located

time series forecasting relies on the strong dependencies along both spatial and temporal axis. For instance, the traffic flow around the commercial center evolves during the day (*temporal axis*) and is impacted by the traffic congestion from nearby areas and the competition from the rival commercial centers (*spatial axis*). The widely accepted fact is that the spatial correlation significantly affects forecasting performance. Recent studies [237, 130] have shown that models that explicitly account for the underlying relationships across multiple MTS outperform models that forecast each TS in isolation. How to model the spatial correlation has become a fundamental research problem that recent studies are dedicated to solving, especially in the contexts where the data has strong spatial dependencies, such as forecasting traffic data, air pollution, weather, etc.

There are mainly two classes of work for geo-located time series forecasting: *graph-based* and *non graph-based* models. The latter considers the spatial network as a graph and adopts the graph neural networks (GNNs) [259] for modeling the spatial interactions. We show a taxonomy of the related forecasting approaches in Figure 2.7.

2.3.2.1 Non Graph-based Models

The general MTS forecasting models can be adopted for geo-located time series forecasting. A huge amount of work [128, 260, 132, 134] have conducted their experiments on the Spatio-temporal datasets. However, they generally ignore the spatial information in the data to conduct the general time series forecasting. As mentioned previously, when integrating spatial information, the model is capable of combining rich external information to improve its performance. A powerful family of approaches [261, 262, 256, 263, 264] adopt the spatial information by converting the spatial network into a grid-based representation as shown in Figure 2.8. The geo-located time series data at each timestamp can be projected into a data matrix, thus converting the data into a matrix sequence while conserving the spatial information.



(a) Grid-based segmentation **(b)** Traffic inflow matrix

Figure 2.8: (Figures taken from [261]) Grid-based data representation of the traffic flow data in Beijing city: (a) the map segmentation of the spatial network, (b) the traffic inflow matrix at one single time point.

With the grid-based representation of the geo-located time series, a local CNN module can be applied to extract the spatial information at each timestamp which can be combined with other temporal units (e.g., RNN) to output the Spatio-temporal representations of the geo-located time series. Various work [256, 257, 261, 262, 263, 264, 265, 266, 267, 268, 269] designed their models based on this methodology and consider the characteristics of geo-located time series from different aspects. For instance, ConvLSTM [256] replaces the dot product in LSTM (Long Short-Term Memory) by a convolutional operation in both the input-to-state and state-to-state transitions, which better models the local spatial features than a fully-connected LSTM. Similar to ConvLSTM, TrajGRU [262] learn the *location-variant* structure for the recurrent connections in GRU (Gated Recurrent Unit) where the spatial correlations can be dynamic at different time stamps. DMVST-Net [264] propose to use a multi-view framework to model both spatial and temporal relations. Specifically, it consists of three views: temporal view (modeling correlations between future values with near time points via LSTM), spatial view (modeling local spatial correlation via local CNN), and semantic view (modeling correlations among regions sharing similar temporal patterns). The three views can be combined via a Fully Connected (FC) layer to output the Spatio-temporal representation. Instead of modeling the temporal decency via an RNN-like module, various work combines directly the Spatio-temporal features extracted from each timestamp. ST-ResNet [261] applies a ResNet [118] architecture with CNN blocks on each of the data matrices with more attention to the recent observations. The output of each timestamp is merged with the contextual features as the Spatio-temporal representation. Based on ST-ResNet, DeepSTN+ [263] further models the long-range spatial dependence among the geo-located TS in different regions. A two-channel structure for extracting the spatial features is adopted. One CNN-based channel extracts the local (short-range) spatial relationship; another fully-connected channel captures long-range spatial dependence.

2.3.2.2 Graph-based Models

The spatial network in geo-located time series can be essentially considered as a graph. In recent years, graph neural networks (GNNs) have been introduced and have achieved state-of-the-art performance in a series of Spatio-temporal forecasting problems [250]. Given a spatial network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, ..., v_N\}$ is a set of N spatial nodes and $\mathcal{E} = \{e_1, ..., e_E\}$ is a set of E edges connecting the nodes, the spatial topology construction in GNNs is an essential step for building the model. In general, the graph (network) can be converted into a $N \times N$ adjacency matrix containing the edge weights between the node pairs. The edge weight reflects the spatial interactions between nodes i and j. Both the nodes and the edges represent different attributes in different GNN problems. In the literature, various graph structures are designed to model the geo-located time series forecasting problem, which can be divided into two classes: *pre-defined graphs* and *learnable graphs*.

For pre-defined graphs, the adjacency matrix can be either connection-based or distance-based. First, the connection-based matrix is commonly used to represent the connectivity

between the spatial nodes. The matrix has a binary format, with an element value of 1 if connected and O otherwise. A typical example is related to the road network in the Spatio-temporal traffic data. The connection-based matrix is adopted where two regions are reachable by different modes of transport, such as motorway, highway, or subway. Various work builds the GNNs based on the connectivity matrix. For instance, ASTGCN [270] applies an attention mechanism on both spatial and temporal dimensions. Specifically, a graph convolution is involved in the spatial dimension, capturing spatial dependencies from the neighborhood. A temporal convolution is applied along the temporal dimension, exploiting temporal dependencies from nearby time points. Instead of using separate components to capture spatial and temporal correlations, STSGCN [271] captures the localized Spatio-temporal correlations in heterogeneous Spatio-temporal data. This Spatio-temporal correlation is further enhanced by STFGNN [272] which measures the similarity between spatial nodes to enhance the spatial correlations. The similarity is calculated with DTW distance on the time series data of different locations. Second, the widely used distance-based matrix represents the spatial closeness between the graph nodes. It can be applied in the context where there is no evident connection between the spatial nodes, such as air pollution or weather conditions in different areas. A huge amount of work [237, 253, 273, 274, 130] belong to this category. For instance, DCRNN [237] considers the directed distance between the traffic node pairs and applies diffusion convolutions with the adjacency matrix for each direction, a GRU module is applied to capture the temporal dependency; STGCN [130] considers the absolute (undirected) distance between the nodes and applies a temporal CNN for capturing the temporal features; ST-Metanet [253] adopts the contextual features and the distances as meta-data for adding attentions to weight the graph network; Similarly, GMAN [273] applies attention mechanism to weight the Spatio-temporal features but with attention scores calculated on the time series data themselves; ST-GRAT [274] utilizes the spatial and temporal self-attention in a transformer-based model.

Instead of using the pre-defined graph, which requires prior knowledge about the spatial data, we can also consider the spatial structure as something to be learned. Recent studies [275, 276, 277] show that the cross-region dependence does exist for those nodes which are not physically connected or physically far apart but share similar patterns. The learnable graphs can be either static or dynamic. The static graphs are commonly studied for Spatio-temporal forecasting in the literature [278, 275, 276, 279, 280]. For instance, based on DCRNN, Graph WaveNet [278] learns another adjacency matrix with the predefined graphs and integrates diffusion graph convolutions with temporal convolutions; MTGNN [276] adopts a learnable graph and integrates mix-hop propagation layers in the graph convolution module. Moreover, it designed the dilated inception layers in temporal convolutions; AGCRN [279] learns an adaptive graph and integrates with recurrent graph convolutions with node adaptive parameter learning; GTS [280] learns a probabilistic graph which is combined with the recurrent graph convolutions to do traffic forecasting. Different from the static graph, which considers the spatial correlations between the nodes are fixed, one can imagine the dynamic interactions between the spatial nodes. Therefore, the adja-

cency matrix should be dynamic along with time. For instance, in traffic forecasting tasks, the complex relations between roads and vehicles in the spatial dimension play important roles. The temporal state of each node has an impact on other nodes in the network. For instance, the nodes around the entertainment venues and working places have different effects on each other during the working and leisure hours, that is *Dynamic Spatial Correlations*. Recent work shows that learning dynamic spatial correlations brings better model performance than static correlations. For instance, ASTGNN [281] employs the self-attention calculated from the dynamic data to adjust a static adjacency matrix. In this manner, the graph is continuously learned with the dynamic data. Similarly, DGCRN [277] designed the dynamic filters from the dynamic node measurements to adjust a predefined static graph.

2.4 Conclusion

In this chapter, we reviewed the literature of the general time series mining activities with the extension to broader relevant research domains, such as semi-supervised learning and data streams. The time series representation plays a critical role in almost all time series mining tasks, such as classification, clustering, forecasting, etc. We build a taxonomy for the time series representation approaches in the literature, which can be applied separately or in a combined manner, depending on the requirements of the concrete context. We present as well two classic time series mining activities: *time series classification*, *geo-located time series forecasting*, and have reviewed the previous work on learning the representations for such activities.

Time series mining is an extensive research domain with countless real-life applications. A good understanding of the time series' characteristics helps us construct or learn a robust time series representation for various downstream learning tasks. However, in most complex real-life contexts, the cross-domain research based on time series is always envisaged. The real-world applications enrich the time series with more characteristics and apply it to various complex contexts. The interdisciplinary domain raises more requirements on modeling time series for specific tasks. For instance, the time series can be learned in a dynamic streaming context, where the time series representation should be dynamic with time. It is practically common as most time series sources (i.e., IoT sensors) generate data continuously and infinitely. **An intersection between time series and data stream learning** can be considered for such a context; the time series can be multivariate and not fully annotated. In the real world, the time series data is usually collected synchronously from multiple sources, i.e., *multivariate time series* (MTS). The MTS generally requires a considerable annotation effort for human experts as the real-valued sequence is not as interpretable as the classic data (e.g., image, text). Therefore, **an intersection of MTS analysis and semi-supervised learning** happens in this context; the MTS can be collected from multiple sources on different spatial locations, the time series data then becomes geo-located, i.e., geo-located time series. The representation

learning on such domain-specific time series should consider not only the concrete learning task (e.g., forecasting) but also the factors which impact the task, such as the spatial information, the imperfect data, etc. **An intersection between time series forecasting and spatial/imperfect data analysis** can be considered.

Many more intersections between time series learning and other research domains exist in real life, as the time series data is widely available in real-world applications but under various contexts with distinct requirements. In the following chapters, we present our contributions on three interdisciplinary studies: (i) (**Chapter** 3) time series representation learning in streaming context (*time series & data stream*); (ii) (**Chapter** 4) multivariate time series learning in a weakly supervised manner (*time series & semi-supervised learning*); (iii) (**Chapter** 5) geo-located time series forecasting with missing values (*time series & spatial data & imperfect data*).

Chapter 3

Dynamic Feature Learning on Time Series Stream

Contents

3.1 Introduction

Time Series features, or Time Series representation learning is a classic research problem with enormous real-life applications. To explore the supervised features in Time Series, we usually adopt the classification as the learning task. Time Series Classification (TSC)

is intended to predict the label of a newly input TS instance by extracting the knowledge from collected data. Various TSC approaches have been proposed by researchers in recent years which are suitable for different contexts along with dissimilar TS features. One Nearest Neighbor (*1-NN*) classifier for whole series similarity measure is a typical baseline of TSC research, which is usually combined with various distance measures [189, 190, 188, 191]. Instead of considering the global feature of entire series, summary statistic features (e.g., mean, deviation, slope, etc.) can be extracted from every sub-series to build diverse ensemble classifiers [192, 193, 194]. With the emergence of TS dimensionality reduction techniques (e.g., PAA [45], SAX [86], etc.), TS instances can be represented by high-dimensional vectors so that various techniques [77] from classic data analysis can be adapted into TS context. For instance, in case that motifs or frequent patterns are what characterize a given class, the dictionary based approaches [9, 91, 282] borrowed from Text Mining and Information Retrieval community can be adopted. As for the scenario that the occurrence of specific sub-series determines a class, TS can then be represented by such shape-based features, namely Shapelet [14]. Various Shapelet-based approaches have been proposed to optimize both the accuracy [97, 99] and the efficiency [209, 100] of the classification. Another remarkable attempt [6, 7, 195] adopting ensemble approaches on several TS representations (e.g., Shapelet-based, similarity-based, interval-based etc.) shows a superior accuracy to one single representation classifiers, where TS features are from different representation domains, and can not be presented in a single form.

The optimization of TS feature extraction and model construction process allows us to strive for a low prediction error, and stay as close as possible to Time Series' nature Concept [283], which refers to the target variable that the learning model is trying to predict. Most *TSC* approaches are biased towards learning from an off-line Time Series dataset, with the assumption that data instances are independently and identically distributed (i.i.d) within a particular concept, but rarely consider the streaming context, where a gradual change of the concept happens along with the input of TS stream, that is *Concept Drift*. For instance, the most accurate ensemble classifiers [6, 7] are not good options in streaming context due to their complex architecture. Lazy classifiers on Time Series such as Nearest Neighbor (1-NN) [8] and dictionary based approaches [9] are applicable for streaming context. However, every input instance will be considered to adjust the inner concept, which requires potentially a large buffer space and will bring a huge computation cost. Recent Deep Neural Network (DNN) approaches [3, 10, 11, 12] on TSC are capable of tuning the model incrementally, but stay always in an awkward position for the lack of explainability, which is required by domains like healthcare where questions of accountability and transparency are particularly important.

Shapelet, as a shape-based feature in TS, which is widely adopted by the community for its reliability and interpretability, provides a possibility to fulfil the aforementioned requirements. However, the high degree of coupling inside classical Shapelet extraction algorithm [14] leads to the problem of not being able to parallelize. Even though some speed-up techniques such as Early Abandoning [14] are proposed, the Euclidean Distance-

based algorithm for extracting the Shapelets remains resource costly with a time complexity of $\mathcal{O}(N^2n^4)$.

To fill the gap between Time Series Classification and data streams processing, in this chapter, we propose a TS stream learning framework, where TS features and models can be updated with consideration of Concept Drift. First, we propose an offline algorithm SMAP, which allows extracting the Shapelet features in a distributed manner under Spark framework while conserving the model's interpretability. Then, we propose an incremental version of SMAP, namely ISMAP, allowing us to further explore the *Test-then-Train* strategy, to evaluate the learning model constantly on newly input instance, then update the model regarding to the evaluation result. The cached information under old concept will be eliminated gradually by an elastic caching mechanism, which deals with the challenge of infinite streaming instances.

The rest of this chapter is organized as follows. First, we give the necessary notations used in this chapter. Then, we review the state-of-the-art approaches which are useful for our research problems. We describe later our proposals: (i) SMAP for extracting Shapelet in an offline manner; (ii) ISMAP for extracting Shapelet online in both TS stream contexts of stable concept and drifting concept. Next, we shows an empirical evaluation of our proposals on real-life datasets. Finally, we give our conclusions and perspectives for future work.

3.2 Background and State-of-the-art

3.2.1 Definitions and Notations

We start with defining the notions used in the chapter:

Definition 3.1. (Time Series). *Time Series* T is a sequence of real-valued numbers T=(t_1, t_2, ..., t_i, ..., t_n), where n is the length of T.

Definition 3.2. (Time Series Stream). *Time Series Stream* S_{TS} is a continuous input data stream where each instance is a Time Series: S_{TS}=(T_1, T_2, ..., T_N, ...). Notice that N increases with each new time-tick.

Definition 3.3. (Time Series Chunk). *Time Series Chunk* $C_{t,w}$ is a Time Series micro-batch at time-tick t with window size w in S_{TS}: $C_{t,w}$=(T_{t-w+1}, T_{t-w+2}, ..., T_t).

Definition 3.4. (Cached Dataset). A *Cached Dataset* D_t is a set of time series T_i, and class label c_i, collected after time tick t. Formally, D_t= $\langle T_t, c_{j_t}\rangle$, $\langle T_{t+1}, c_{j_{t+1}}\rangle$, ..., $\langle T_N, c_{j_N}\rangle$, where N is the time-tick of the most recent input instance. $C = c_1, c_2, ..., c_{|C|}$ is a collection of class labels, where $|C|$ denotes the number of labels.

Definition 3.5. (Subsequence). A *subsequence* $T_{i,m}$ of Time Series T is a continuous subset of values from T of length m starting from index i: $T_{i,m} = (t_i, t_{i+1}, ..., t_{i+m-1})$, where $i \in [0, n-m+1]$.

Definition 3.6. (Shapelet). *Shapelet* $\hat{s}$ is a time series subsequence which is particularly representative of a class. As such, it shows a shape which can distinguish one class from the others.

Definition 3.7. (Z-Normalized Time Series). *Z-Normalized Time Series* is a formal representation of Time Series, which is defined as:

$$Normal(T) = \frac{T - \mu}{\sigma} \tag{3.1}$$

where μ is the sample mean, σ is the standard deviation:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} t_i, \quad \sigma^2 = \frac{1}{n}\sum_{i=1}^{n} t_i^2 - \mu \tag{3.2}$$

Z-Normalization allows us to focus on the structural feature of T, rather than its amplitude value. It addresses the problem of data stability. For instance, assume that the Euclidean Distance(ED) between two time series T, T' of the same length n is expressed as follows:

$$ED_{T,T'} = \sqrt{\sum_{i=1}^{n}(t_i - t_i')^2} \tag{3.3}$$

Some little changes (e.g., the noise) will cause an evident bias for the result, Z-Normalization is a way of smoothing the bias value.

3.2.2 Time Series Feature Representations

Various TSC approaches have been proposed by researchers in recent years which are suitable for different contexts along with **dissimilar TS features**:

- The global feature of entire series [188] for One Nearest Neighbor (*1-NN*) classifier, which is usually combined with various distance measures [189, 190, 188, 191].
- The summary statistic features (e.g., mean, deviation, slope, etc.) extracted from every sub-series to build diverse ensemble classifiers [192, 193, 194].
- Pattern [91] features when the specific patterns or the pattern combinations characterize a class, including but not limited to PAA [45], or SAX [86] based TS dimensionality reduction techniques [77].
  - in case that the frequent patterns are what characterize a given class, the dictionary based approaches [9, 91, 282] borrowed from Text Mining and Information Retrieval community are usually adopted.

  - if the occurrence of specific sub-series determines a class, TS can then be represented by such shape-based features, namely Shapelet [14]. Various Shapelet-based approaches have been proposed to optimize both the accuracy [97, 99] and the efficiency [209, 100] of the classification.

- Deep representation features: the last layer of a neural network classifier [3] embeds the TS features, which lack interpretability and are hard to be explored.

- Feature combinations (e.g., global feature-based, frequent motif-based, Shapelet-based, etc.) via ensemble approaches [188, 7, 195]. Despite of a superior accuracy to one single feature representation classifiers, the ensemble approaches focus more on the learning model but not the features themselves.

In this work, we focus our attention on the *Shapelet* [14] feature, which provides an interpretable way to show the learned representations under both offline and online context.

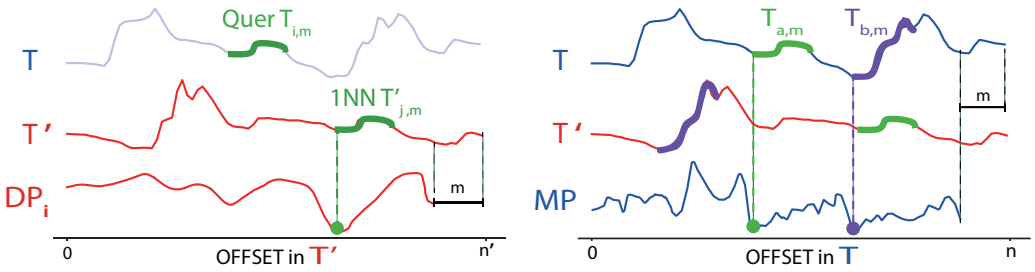
3.2.3 Matrix Profile in Time Series Mining

Matrix Profile, firstly proposed in [13], has be becoming a popular technique for generalising the time series mining problems into a unified processing framework. It has implications for time series motif discovery, time series joins, shapelet discovery (classification), density estimation, semantic segmentation, visualization, rule discovery, clustering, etc. All the downstream mining tasks are based on the two following concepts:

Definition 3.8. (Distance Profile). *Distance Profile* DP_i is a vector which stores the Euclidean Distance between a given subsequence/query $T_{i,m}$ in source T and every subsequences $T'_{j,m}$ of target T'. Formally, $DP_i^m = (DP_{i,1}^m, ..., DP_{i,j}^m, ..., T_{i,n'-m+1}^m)$, where $DP_{i,j}^m = dist(T_{i,m}, T'_{j,m}), \forall j \in [0, n' - m + 1]$, n' is the length of T'.

Each element in DP_i is calculated by Euclidean Distance between z-normalized subsequences [16]. From **Fig.** 3.1*(a)*, we can visually obtain the position of Query's Nearest Neighbor (1NN) in T' from the lowest point in DP_i. In general, we slide the window of size m in the target T' to obtain the Euclidean Distance between the query and target subsequence, leading to a time complexity of $\mathcal{O}(nm^2)$ to compute the distance profile.

Authors in [13] propose *MASS* which is considered as the fastest exact distance measure between two Time Series. *MASS* computes Distance Profile based on Fast Fourier Transform (FFT), which requires only $\mathcal{O}(nlogn)$ time and is independent of query's length, instead of $\mathcal{O}(nm^2)$ [14] by classical sliding window measure.



(a) Distance Profile DP_i **(b)** Matrix Profile MP

Figure 3.1: (a) Distance Profile between Query $T_{i,m}$ and target time series T', where n' is the length of T'. Obviously, $DP_{i,j}$ can be considered as a meta TS annotating target T'; (b) Matrix Profile between Source time series T and Target time series T', where n is the length of T. Intuitively, MP_i shares the same offset as source T

Definition 3.9. (Matrix Profile). *Matrix Profile* MP is a vector of distance between every subsequence $T_{i,m}$ in source T and its nearest neighbor $T'_{j,m}$ in target T'. Formally, MP^m $=(MP_1^m, ..., MP_i^m, ..., MP_{n-m+1}^m)$, where $MP_i^m = min(DP_i^m)$, $i \in [0, n-m+1]$, n is the length of T.

Unlike the distance profile, the matrix profile is a meta TS annotating the source TS. As shown in **Fig.** 3.1*(b)*, the lowest point in MP show the position of query which has the most similar matching in target TS.

3.3 Problem Statement

First, the previous work for extracting Shapelets [14] is along with a high degree of coupling , leading to the problem of not being able to parallelize. The speed-up method such as Early Abandoning [14] is based on classical Euclidean Distance measure, which has a time complexity of $\mathcal{O}(N^2n^4)$ with several orders of magnitude higher than *MASS* [13]: $\mathcal{O}(N^2n^3logn)$. Another common trick played by previous work [14]: If we know $\hat{s}$ is a low-quality candidate, then any similar subsequence $\hat{s}'$ to $\hat{s}$ must also result in a low quality and therefore, a costly computation of the distance set $D_{\hat{s}'}$ (evaluation of $\hat{s}'$) can be skipped. However, a candidate shapelet is evaluated by its quality ranking among all candidates of the same length. Assume that the distributed nodes have generated from dataset a collection of candidates $\hat{s}_l$, an aggregation operation between nodes is required to extract the candidate with the best quality. Extra aggregations will be made along with the iteration of candidate length. Apparently, the acceleration from the classical pruning techniques can be easily offset by the communication cost caused by the aggregation.

Second, even though the Shapelet-based approaches allows learning an interpretable representation for *TSC* task, they are limited to the off-line learning from a constant Time Series dataset, in which the data instances are independently and identically distributed

(i.i.d) within a stable concept. The real-life streaming context raises more challenges for learning the interpretable representations: (i) the Shapelet-based models are generally not incremental with the newly input instances, in other words, the model needs to be re-trained once the database is updated; (ii) the models are not eligible to consider the gradual change of the concept along with the streaming instances, i.e., *Concept Drift*; (iii) with limited storage space, the models are not applicable when the streaming instances are infinite.

3.4 Our proposals

In this section, we present our proposals under both offline and online mode for extracting TS features for downstream classification tasks. We start by introducing the offline SMAP algorithm. With the help of the Big Data framework (e.g., Spark), SMAP allows extracting the Shapelet features in a distributed manner based on the Matrix Profile concept. Then, we describe the online framework ISMAP, which learns the dynamic TS representations in the streaming context.

3.4.1 Shapelet extraction on MAtrix Profile (SMAP) for TSC

The main idea of our system is that the calculation should be shared and executed independently, less communication between the nodes, more powerful the algorithm would be. The conventional Time Series classification problems are usually tackled with nearest neighbor (kNN) algorithm [14] due to its easy-design feature. The processing of labelled Time Series data requires to be flexibly arranged for nodes in the cluster, where the executors share the CPU/memory resource. To this end, a suitable algorithm is applied here allowing assignment of computing tasks which are relatively independent of each other.

3.4.1.1 General Idea

We start by introducing two extended concepts of Matrix Profile which are proposed for extracting features for TS classification:

Definition 10: *Representative Profile* RP_T^C is a vector of representative power of subsequences in T for class C: $RP_T^C = (RP_{T_1}^C, ..., RP_{T_i}^C, ...RP_{T_{n-m+1}}^C)$

The representative power of subsequence T_i in class C is defined as:

$$RP_{T_i}^C = avg(MP_{T_i,T'}) \tag{3.4}$$

where $T' \in D_t^C$. Intuitively, $RP_{T_i}^C$ is a normalized distance between T_i and global TS instance cluster of class C, it represents the relevance between the subsequence T_i (i.e., the candidate Shapelet) and the class. As shown in **Fig.** 3.2 (b)(d), a threshold can be set to show the starting index area in T, where the subsequences are representative for class C.

Definition 11: *Discriminative Profile* $DiscmP_T$ is a vector of discriminative power of subsequences in T:

$$DiscmP_T = RP_T^{NonC} - RP_T^C \tag{3.5}$$

The discriminative power of T_i in dataset shows the difference of representative power of candidate Shapelet from its own class to the others *(OVA, one-vs-all)*, which follows similar heuristics in [284], where authors proved *OVA* strategy in Shapelet quality assessment performs better than traditional Decision Tree approach [14] in both accuracy and efficiency metrics. Intuitively, Discriminative Profile can give a global view of the important patterns' positions over a Time Series. As shown in **Fig.** 3.2 (f)(g), the highest point in the profile shows the position of the sub-series in T, which has the biggest skewing of relevance between class C and other classes. Through setting a power threshold, the discriminative pattern, that is the candidate Shapelet, can be visually identified in T by their discriminative power.

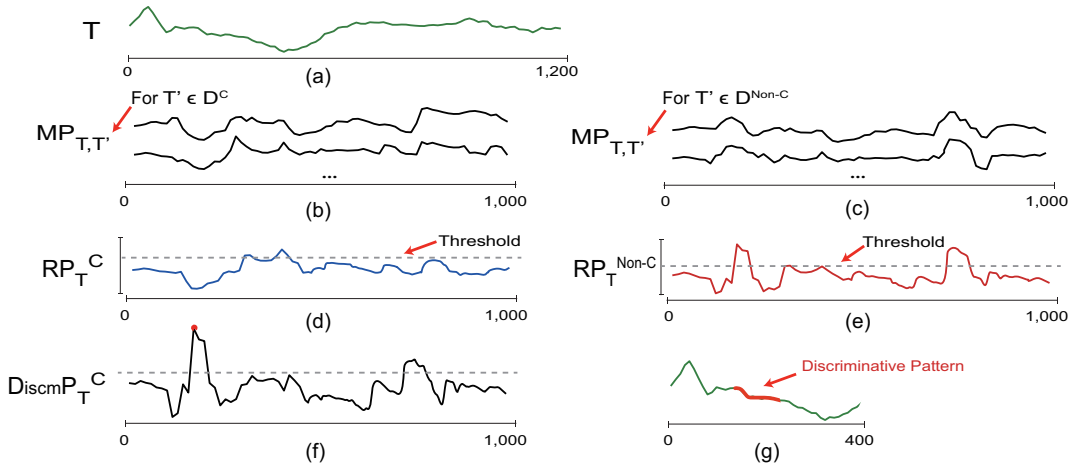


Figure 3.2: *(a)* Time Series T with class C. *(b,c)* Matrix Profile set for T with TS instances in different classes. *(d,e)* Representative Profile of T in different classes. A threshold can determine the representative area in T. *(f)* Discriminative Profile of T in dataset. The highest point in $DiscmP_T^C$ identifies the most discriminative pattern's in T.

Matrix profile [13] provides a meta-data which facilitates the representation of a complex correlation between two time series. With the help of the *Discriminative Profile*, we can then identify the candidate Shapelets in each TS instance as well as their discriminative power of the classes.

3.4.1.2 Distributed Algorithm Design

As shown in *Algorithm* 1, considering time series as the smallest processing unit between Spark nodes, *SMAP* firstly broadcasts the dataset to distributed nodes (*line 3*) in order to reduce the communication cost from repetitive access of common data. Then, each cluster partition shares the computing tasks for a set of TS (*line 4*), and extracts the most

discriminative sub-series of various length in each processing unit (*line 7*). Each extracted sub-series can be considered as a candidate Shapelet, which is assigned a distance threshold defined by its representative power in its own class. To put it simply, the representative power of a subsequence in class C, is its normalized distance to the global instance cluster of class C. Intuitively, it represents the relevance between the subsequence (i.e., the candidate shapelet) and the class. Therefore, the threshold can determine the inclusion between the candidate Shapelet and a TS. A strategy to check if T contains a candidate Shapelet $\hat{s}$ can be defined as the following:

$$Inclusion(T, \hat{s}) = \begin{cases} true, & if\ dist(T, \hat{s}) \leq \hat{s}.dist_{Thresh} \\ false, & otherwise \end{cases} \tag{3.6}$$

A quality *Normalization* in *line 8* is made which allows to assess the discriminative power for the candidate shapelets of different length in an uniform way. Similar as the concept *Information Gain* [14], but discriminative profile is a technique more interpretable serving to assess the candidate shapelets. Moreover, in this manner, a threshold distance for deciding the TS-Shapelet inclusion can be given directly, instead of iterating every possible distance and deciding the best one with the highest Information Gain, in time $\mathcal{O}(N^2n^2)$.

Each TS unit is assigned an unique Hash ID to reduce the volume of transferred data between nodes. The TS ID, as well as the discriminative power and threshold distance of its contained candidate Shapelets, will be output as the computing results of the partition (*line 10*). Finally, a single aggregation process between nodes (*line 11*) is required to obtain the Shapelet result of different classes. As shown in **Fig.** 3.2, the whole extraction process can be visualized with a strong explainability, and generates high interpretable results.

3.4.1.3 Optimization Strategy

The pruning function in *line* 9 is capable of eliminating the number of candidate shapelet, and then reducing the communication cost during the aggregation process. We can simply take the "TopK" strategy, which extracts the biggest K values of the *Discrimination Profile*. However, such a technique is far from lightening the computation during *MapPartition* process.

Since each processing unit should be independent from each other, a tenable technique for updating the profile of a long range query could be adopted. The *Lower Bounding distance* [285] is defined to estimate a minimal possible Z-Normalized Euclidean Distance between two subsequences $T_{i,l+k}$ and $T_{j,l+k}$, based on the distance already computed between $T_{i,l}$ and $T_{j,l}$. Compared to a linear time complexity of computing the exact distance, LB Distance Profile can be calculated in a constant time, which can accelerate greatly the computation of Matrix Profile in *Figure 4*. For example, from shapelet length $l = m$ to $m + 1$, the time complexity of computing the distance $d_{i,j}^{l+1}$ is $\mathcal{O}(n\frac{m(m-1)}{2}m)$, where

Algorithm 1: SMAP (Shapelet extraction on MAtrix Profile on Spark)

Input: Dataset D, **classSet** $\hat{C}$, k
Output: $\hat{S}$

$l_{min} \leftarrow 0.1 * getMinLen(D)$, $l_{max} \leftarrow 0.5 * getMinLen(D)$,
$DiscmP \leftarrow []$, $dist_{Thresh} \leftarrow []$, $\hat{S} \leftarrow \emptyset$
D.broadcast(); //each TS has an unique ID
MapPartition $\underline{(Set\ of\ \langle ID, T\rangle : T_{set})}$
  for $\underline{\langle ID, T\rangle \in T_{set}}$ **do**
    for $\underline{m \leftarrow l_{min}\ \text{to}\ l_{max}}$ **do**
      $DiscmP[m], dist_{Thresh}[m] \leftarrow computeDiscmP(T, D, m)$
      $DiscmP[m] \leftarrow DiscmP[m] * \sqrt{1/m}$
    $DiscmP, dist_{Thresh} \leftarrow pruning(DiscmP, dist_{Thresh})$
    emit$(ID, DiscmP, dist_{Thresh})$
MapAggregation $\underline{(class, (ID, DiscmP, dist_{Thresh}))}$
  for $\underline{c \in \hat{C}}$ **do**
    $\hat{S}' \leftarrow getTopk(DiscmP[c], dist_{Thresh}[c], k)$
    $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
return $\hat{S}$

$j \in [0, n\frac{m(m-1)}{2}]$ which represents the number of subsequences in D, n is the number of instance in D, m is the length of the longest instance in D. Accordingly, LB distance takes $\mathcal{O}(n\frac{m(m-1)}{2})$ which shows an apparent advantage when the query length is relatively long. Lower Bounding distance [285] is defined as:

$$LB(d_{i,j}^{l+k}) = \begin{cases} \sqrt{l}\frac{\sigma_{j,l}}{\sigma_{j,l+k}}, & if q_{i,j} \leq 0 \\ \sqrt{l(1-q_{i,j}^2)}\frac{\sigma_{j,l}}{\sigma_{j,l+k}}, & otherwise \end{cases} \tag{3.7}$$

where $q_{i,j} = \frac{\sum_{p=1}^{l}\frac{(t_{j+p-1}t_{i+p-1})}{l} - \mu_{i,l}\mu_{j,l}}{\sigma_{i,l}\sigma_{j,l}}$

Empirically, the matching subsequence $T_{j,l}$ which is the nearest neighbor of $T_{i,l}$, can deduce a longer subsequence $T_{j,l+1}$, which is probably the nearest neighbor of $T_{i,l+1}$. Assume that the matching subsequence keeps in the same position in T_{target} when query $T_{i,l}$ length increases, then the time complexity for computing the minimal distance between $T_{i,l}$ and T_{target} is $\mathcal{O}(l)$, other than $\mathcal{O}(l(n-l+1))$. As mentioned previously, $MP_i^m = min(DP_i^m)$, the main idea here is to utilize LB Distance to accelerate the computation of $min(DP_i^m)$, rather than computing the entire DP_i in a higher time complexity.

To sum up this section, Discriminative Profile provides a possibility to extract the interpretable patterns in an explainable manner. The adoption of *MASS* when computing the distance profile essentially accelerates the extraction process compared to using pruning techniques based on brute force approach[14]. *SMAP* provides a parallel processing mechanism to conduct the extraction in a minimum communication cost on Spark cluster. In the next section, we will show an advanced algorithm which, when applied on Spark cluster, is capable of updating Shapelet results by adopting an incremental model in dynamic source context.

3.4.2 Incremental SMAP (ISMAP)

In this section, we start by studying the incrementality of SMAP, which is a necessary condition for learning in streaming context. Then we propose the evaluation strategies to accelerate incremental learning process and adapting it to streaming context considering Concept Drift.

Typically, a non-incremental algorithm requires to re-pass the existing dataset and conduct a large amount of redundant computations. In *Algorithm* 2, we show Incremental Shapelet extraction on Matrix Profile on Spark (*ISMAP*), which avoids essentially the repetitive computations on existing dataset. As in Spark environment, the communication cost between distributed nodes is a key factor of system's efficiency. The computing task in each Spark partition should be relatively independent without frequent exchange of intermediate results with other partitions. In light of this, we need to make use of the parallel mechanism to well manage the allocation of computing tasks.

As shown in *Algorithm* 2, we assume that each Spark partition keeps a set of Time

Algorithm 2: ISMAP(Incremental Shapelet extraction on MAtrix Profile on Spark)

Input: **Partition** $[ID, T, DiscmP, distThresh]$, **New input** T_N, **classSet** $\hat{C}$, k
Output: $\hat{S}$

$l_{min} \leftarrow 0.1 * getMinLen(D)$, $l_{max} \leftarrow 0.5 * getMinLen(D)$,
$DiscmP \leftarrow []$, $distThresh \leftarrow []$, $\hat{S} \leftarrow \emptyset$
$\langle ID_N, T_N \rangle$.broadcast();
MapPartition $([ID, T, DiscmP, distThresh])$
    / 1. compute the Matrix Profile between T_N and all TS in dataset */*
    / 2. update the current DiscmP of all TS in dataset */*
    / 3. prepare MP_{T_N} elements to compute $DiscmP_{T_N}$ */*
    for $m \leftarrow l_{min}$ to l_{max} **do**
        $MP_T[m] \leftarrow computeMP(T, T_N, m)$
        $MP_{T_N}[m] \leftarrow computeMP(T_N, T, m)$
        $DiscmP[m], distThresh[m] \leftarrow updateDiscmP(DiscmP[m], distThresh[m], MP_T[m])$
    $DiscmP, distThresh \leftarrow pruning(DiscmP, distThresh)$
    emit$(ID, T, DiscmP, distThresh, MP_{T_N})$
MapAggregation $(*, (ID, T, DiscmP, distThresh, MP_{T_N}))$
    $DiscmP_{T_N}, distThresh_{T_N} = computeDiscmP(collect(MP_{T_N}))$
    $DiscmP_{T_N} \leftarrow DiscmP * \sqrt{1/m}$
    $DiscmP_{T_N}, distThresh_{T_N} \leftarrow pruning(DiscmP_{T_N}, distThresh_{T_N})$
    cache$(ID_{T_N}, DiscmP_{T_N}, distThresh_{T_N})$
MapAggregation $(class, (ID, DiscmP, distThresh))$
    for $c \in \hat{C}$ **do**
        $\hat{S}' \leftarrow getTopk(DiscmP[c], distThresh[c], k)$
    $\hat{S} \leftarrow \hat{S} \cup \hat{S}'$
return $\hat{S}$

Series with their Discriminative Profiles and corresponding Threshold Distance sets. The newly input Time Series T_N will be broadcast to each distributed node. Information in T_N should be extracted and merged to existing knowledge base, which can be carried out into two steps:

1. **Update existing Shapelets**: With newly input instance T_N, existing candidate Shapelets should update their representative power in each class, and discriminative power in current dataset.
2. **Evaluate new candidate Shapelets**: T_N will introduce new candidate Shapelets of various length, which should be evaluated and placed into Shapelet ranking list by their discriminative power.

Step (1) is shown in *line 9,11*, from the *Formula* 3.4 and 3.5, we can observe that the linearity of Discriminative Profile makes the fact that each existing TS only need one single Matrix Profile computation with T_N to update the candidate Shapelets. As for Step (2), the Discriminative Profile computing of T_N is shared on different Spark partitions, where Matrix Profiles with existing TS instances are computed in *line 10*, an aggregation process in *line 14-18* extracts the discriminative patterns in T_N, which will be aggregated with existing candidate Shapelets and update the output results in *line 19-22*.

Like classical incremental algorithms, *ISMAP* takes all input instances into account, which means every input TS instance will be imported into the system to update the Shapelet, even if the computing imports no valuable information into the system, that is, the information contained in the instance is repetitive with that in the knowledge base. Evidently, we are capable of avoiding the redundant information's computation by adopting an interleaved *Test-then-Train* strategy [5] with an extra Shapelet evaluation process over input instances.

3.4.2.1 Shapelet Evaluation

The intuition behind the evaluation procedure is that once we have a bad evaluation result, we need to import the instance batch into Shapelet Extraction process, to update the output Shapelet result. As the evaluation time $\mathcal{O}(n - m + 1)$ for a TS instance is much less than that of extraction computing ($\mathcal{O}(Nn^3logn)$), then an evaluation module can improve the system's efficiency by preventing the computation of certain valueless instances. However, how to define that an instance is valueless stays a problem to resolve.

The classical Shapelet-based approach [14] supposes that a Time Series T can be classified by the inclusion of a class-specified Shapelet $\hat{s}$. (i.e. if $dist(T, \hat{s}) \leq \hat{s}.dist_{thresh}$, then $T.class = \hat{s}.class$). The threshold distance of Shapelet gives a split point to decide the TS-Shapelet inclusion. As shown in [208], various approaches (e.g, Information Gain (IG), Kruskal-Wallis (KW) and Mood's Median (MM)) can be applied for both Shapelet assessment and split point decision. Representative Profile and Discriminative Profile achieve the

same effect with these techniques but in a more interpretable manner. Intuitively, we are capable of deciding whether to import a TS instance into Shapelet Extraction process by evaluating its prediction results on current learning model. The Loss Measure is intended to detect the shift between the learning model and the inner concept of data source. In the context of Shapelet, the distance between the learned Shapelets and input instance is able to represent the loss to some extent. Typically, the distance is compared with Shapelets' threshold distance, which derives the *0-1 Loss Function*:

$$L(Y, h(T)) = \begin{cases} 0, & Y = h(T) \\ 1, & Y \neq h(T) \end{cases} \tag{3.8}$$

where

$$h(T) = \begin{cases} C, & if\ dist(T, \hat{s}) \leq \hat{s}.dist_{Thresh} \\ nonC, & otherwise \end{cases}$$

However, by TS-Shapelet inclusion technique, two Time Series with similar distance to a Shapelet may obtain different classes. In addition, a good prediction result of input TS instance with current Shapelets doesn't mean that the instance contains no useful information for adjusting the learning model. The *0-1 Loss Function* analyzes the surface phenomenon of the prediction but ignored the deep information behind the arbitrary split point technique. A loss measured by a crisp *0-1 Loss Function* is then ill-adapted.

When $dist(T, \hat{s}) \leq \hat{s}.dist_{Thresh}$, the prediction result is relatively acceptable. The problem then becomes how to find a balance between time efficiency and TS information checking (i.e., the exhaustive information extraction). The distance between TS and Shapelets describes the shift between real and learned concept, a small distance leads to a reliable prediction result. As the distance measure is usually data-dependent, and the absolute distance value varies with datasets, then a normalized measure describing the shift scale is required. To this end, can we just convert the TS-Shapelet inclusion problem to the possibility that a TS contains the Shapelet?

As extracted Shapelets try to separate one class to others, TSs in different classes tend to be concentrated on the split point, which causes the main error in prediction. Then we assume that $dist(T, \hat{s})$ satisfies Gaussian distribution, as shown in (3.9), the loss can be smoothed by *Sigmoid* function by considering distance distribution. The split point of $\hat{s}$ defines the expectation σ of the distribution.

$$\begin{aligned} L(Y, h(T)) &= \frac{1}{1 + e^{-(x-\sigma)}}, \quad \sigma = \hat{s}.dist_{Thresh} \\ x &= min(dist(T^C, \hat{s})), \quad \hat{s} \in \hat{S}^C \end{aligned} \tag{3.9}$$

As shown in **Fig.** 3.3, the smaller the loss, the greater the possibility that T will contain the Shapelet. Intuitively, a loss threshold Δ can be set by user for *ISMAP* to control the extraction from input instance, and update incrementally the Shapelet to approach the real concept of data source. When Δ is set to 0.5, it has the same effect as *0-1 Loss Function*.

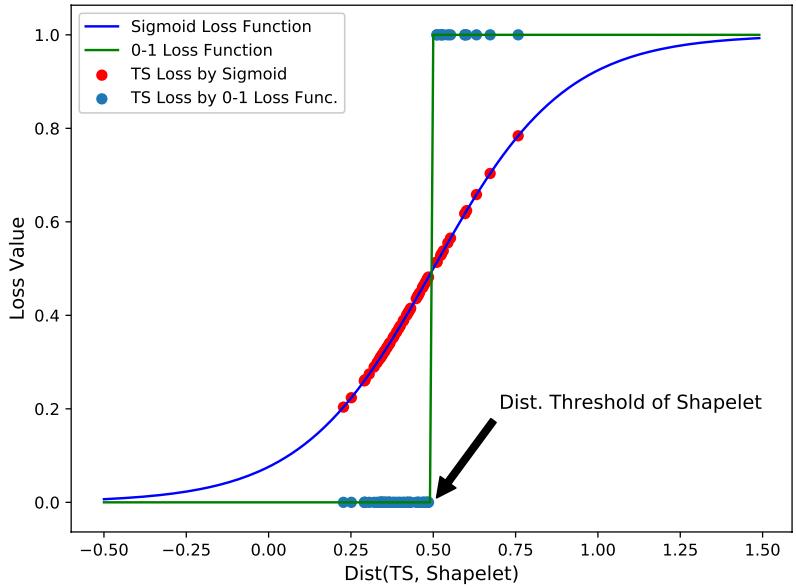


Figure 3.3: Loss measure of Time Series by Sigmoid Function and 0-1 Loss Function, Time Series in different classes are distributed around the split point of Shapelet

However, a stable concept does not hold in several real-life scenarios. For instance, with the soundness of the knowledge in a particular domain, the labeling of newly input instances may evolve gradually, leading to a concept drift. Therefore, the most recent training instances should contribute more than the oldest ones to the target prediction. Then the problem becomes the Concept Drift detection in a Time Series Stream by monitoring the loss function. Conserving the interpretability and explainability of the algorithm, *ISMAP* can be extended to the context of TS Stream by extracting adaptive features.

3.4.2.2 Adaptive feature extraction from Time Series Stream

As shown in **Fig.** 3.4, the system of extracting adaptive Shapelets from Time Series Stream is composed by Shapelet Extraction, Evaluation Bloc and Caching Mechanism. We take TS Chunk $C_{t,w}$ as minimum input unit which contains a number of continuous TS instances: $C_{t,w}$=(T_{t-w+1}, T_{t-w+2}, ..., T_t), where t is the time-tick, w is the window size. By adopting the *Test-then-Train* strategy, the main idea here is to evaluate continuously the shift between learned concept and real concept in data source (i.e., *Test*). Once a Concept Drift is detected, the input chunk will be imported into Shapelet Extraction bloc to update the learning model (i.e., *Train*). Both Shapelet initialization and updating process are parallelizable on Spark cluster, which makes use of RAM as caching unit to lower the I/O cost.

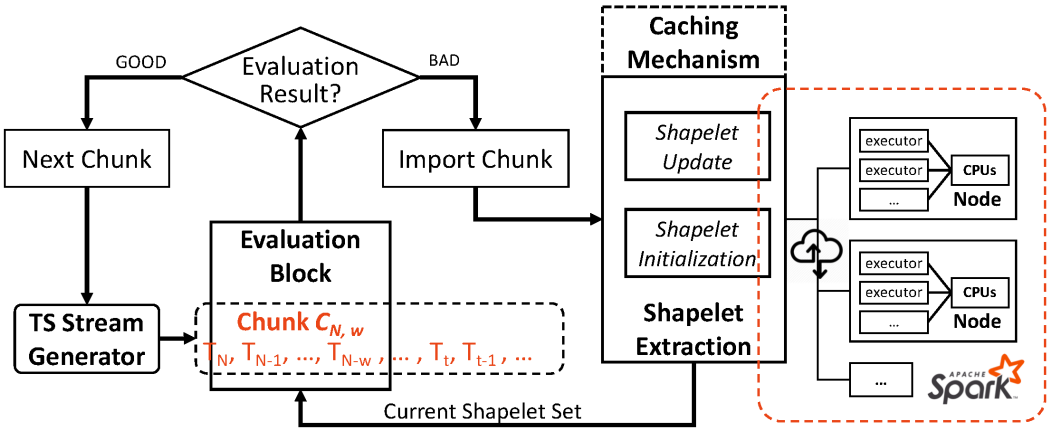


Figure 3.4: System Structure in TS Stream context with Concept Drift

1) Shapelet Extraction:

The computing process follows the same methodology with *ISMAP*, which allows TS instances in the input chunk to be partitioned on various Spark nodes, the discriminative patterns in each partition will be extracted individually and merged between partitions by their ranking power. The Shapelet ranking list is then composed by power-updated existing Shapelets and newly imported candidates.

2) Learned Concept Evaluation:

As aforementioned, the loss of a Shapelet on input instances can describe its shift to real concept of data source. With the same methodology, once a concept drift is detected in data stream, the analysis tends to be more complicated. The challenge here is to distinguish the measured loss from two aspects:

- **Incomplete Extraction:** As main constraint in Shapelet Extraction, insufficient training instances (i.e., under-fitting) will bring a relative high loss. More data will make the learning model approach more the inner concept.

- **Concept Drift:** The measured shift can only reflect the distance to a stable concept, a big shift will be observed using out-of-date learning model.

In light of these challenges, an advanced analysis on detecting the Concept Drift from measured loss is required. That is, not only to measure the loss from each TS Chunk, but also to propose a strategy to analyse the loss. Based on the loss definition in (3.9), we define the average loss for a TS chunk $C_{N,w}$:

$$L_C(N) = \frac{1}{w}\sum_{k=1}^{w} L(Y_{N-w+k}, h(T_{N-w+k})) \tag{3.10}$$

Concept Enrichment: As mentioned in 3.4.2.1, an user-defined loss threshold Δ can be set to decide whether to import the chunk into the system to enrich the concept. That is: $Import_{Chunk} = True$ if $L_C(N) \leq \Delta$.

Concept Drift detection: Page-Hinkley test (PH) [5] is a typical technique used for change detection in signal processing. It allows a loss tolerance for the signal. The sequential test on the variance which considers that normal operation corresponds to a certain variance and a drift being characterized by an increase in this variance. Here we define a cumulative difference between the observed loss and their mean up until the current time:

$$m_N = \sum_{t=0}^{N} (L_C(t) - L_{avg}(t) - \delta) \tag{3.11}$$

where $L_{avg}(t)$ is the average loss until the current time tick t, δ specifies the tolerable magnitude of changes. The minimum value of m_N is defined as $M_N = min(m_t, t = 1...N)$. PH test will measure the difference between M_N and m_N:

$$PH_N = m_N - M_N \tag{3.12}$$

Intuitively, the difference reflects the degree of Concept Drift, when it exceeds a user-specified threshold λ, then the Concept Drift is detected.

3) Caching Mechanism:

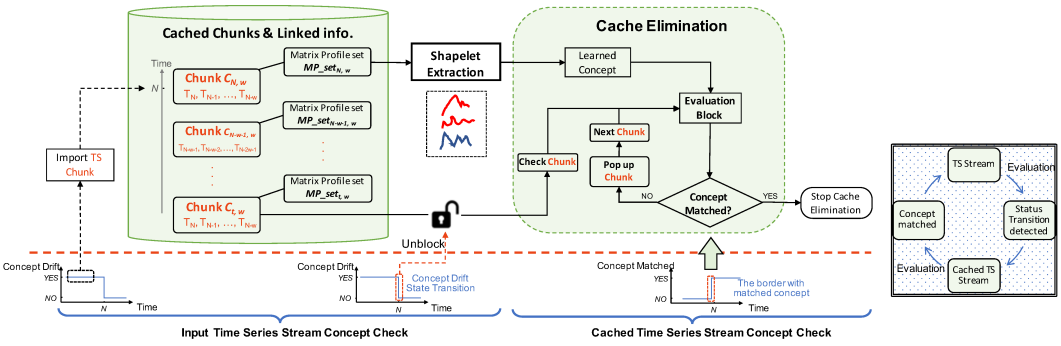


Figure 3.5: Elastic Caching Mechanism for streaming instance chunk in memory

As the discriminative power of a candidate Shapelet is based on its global distribution in dataset, the fact that TS instances should be cached in memory is then a necessary condition of Shapelet Extraction. This is the main difference compared to Concept Drift detection in classical data streams, where it's possible to have one single pass on input instance. Then the main challenge here is to bridge the gap between TS and data stream analysis, that is, to consider the nature of Shapelet in Time Series Stream (i.e., part of data instances need to be cached), and propose a Shapelet-based caching mechanism in streaming context, meanwhile the caching volume should not increase indefinitely along with the never-ending TS Stream.

As Concept Drift is the fact that the prediction targets at different time tick are different, the previous learned concept is inapplicable to current input data. Conversely speaking, the fresh extracted concept doesn't match previous prediction target. This fact opens a path to optimize proactively the data caching procedure in memory. Considered as a complement to the global system shown in **Fig.** 3.4, the caching mechanism is detailed in **Fig.** 3.5. Once a Concept Drift is detected in TS stream, the TS chunk will be cached into memory. As mentioned in *ISMAP*, newly input chunk will generate its Matrix Profile set, and update those of previously cached chunks, which eventually leads to the update of Shapelet list in the learned concept. When there is a state transition in Concept Drift detection, the extraction of a fresh concept is then finished which is applicable for streaming instances coming afterwards. The detection of this transition state triggers then a cache elimination procedure.

The elimination procedure is based on the fact that the prediction target of an old TS Chunk is not compatible with the fresh learned concept. By evaluating the cached chunks chronologically, we aim at finding the transition border where historical chunk starts to match the fresh updated concept, which is a reverse process to the detection of cache elimination trigger. We assume that $C_{t,w}$ is the oldest chunk cached in memory, after a trigger is detected, the evaluation will be conducted from $C_{t,w}$ to more recent chunks using the fresh learned concept. If the prediction target in $C_{t,w}$ matches the fresh concept, that means $C_{t,w}$ is in the frame of the fresh concept, the chunks in later time ticks also contributes to the concept's tuning, which can be kept in memory. Otherwise, $C_{t,w}$ should be removed from cache to eliminate the negative effect to the fresh learned concept. The process will not stop until a transition border is detected. By this proactive mechanism, the system is capable of caching a stable volume of data in TS Stream context, and generating adaptive Shapelets in the frame of drifting concept.

3.5 Experiments and Results

All the programs are implemented under Python 3.6. The source code, as well as the demonstration videos [16, 286] can be found in our project page[1]. The Shapelet Exploration process can be either conducted at local or on a remote Spark cluster. We provide also an 1-click cluster based on Docker, to facilitate the replay of the distributed test offline by the user.

3.5.1 Experimental design

The experiment is conducted by two steps: A). We test the incremental feature and reliability of ISMAP after adopting Shapelet Evaluation process in Test-then-Train strategy. We evaluate the improvement of Shapelet Extraction in both efficiency and accuracy on data

[1]https://github.com/JingweiZuo/TSStreamMining

source with stable concept; B). We check the reliability of adaptive Shapelet Extraction from TS stream with Concept Drift.

In brief, the experiments were designed to answer the following research questions (RQs):

RQ 1 *Incremental learning with ISMAP.* How well our ISMAP model performs in an incremental manner when learning from the stable-concept Time Series Stream?

RQ 2 *Adaptive learning with ISMAP.* How successful is ISMAP in learning the adaptive features in Time Series Stream with concept drift?

-Datasets: UCR Archive [181] is the most complete TS collection in the community, where the datasets are collected from diverse domains, such as readings from Image Outlines, Sensor Readings, Motion captures, spectrograph, and so forth. Each domain matches to certain problem type, which can be best tackled by a specific approach. Authors in [115] studied in detail the approaches and their matching problem types, and pointed out that Shapelet-based method performs relatively better in readings of *Sensor data*, *ECG data*, *Border-converted images*, etc. To this end, we conduct our first incremental experiments on 14 shapelet-characterized datasets in UCR Archive, instead of testing all datasets under various problem types.

However, when we switch to streaming context where Concept Drift happens in TS flow, to the best of our knowledge, currently, the community doesn't collect the dataset which reflects a such phenomenon, due to the fact that the problem hasn't been studied before. Therefore, we generate synthetic datasets by manually adjusting the data source to comply the test scenario. Instead of generating data from scratch to comply Shapelet features, the synthetic data is based on the datasets *Trace* and *ECG5000* (see Table 3.1) from two domains, which have a high reliability in Shapelet-based approaches and eligible for simulating the *Concept Drift* scenario by changing the class within some chunks on different time ticks.

-Data Augmentation: Public datasets, especially those commonly used with shapelets, have relatively small size if we compare them to typical dataset in data mining[2]. A larger number of instances is required for Concept Drift assessment in the streaming TS context. Adding noise is a typical way for data augmentation, as *ISMAP* extracts a range of patterns from each Discriminative Profile and then merge them by their discriminative power, a range-based extraction [287] rather than a single top value is noise resistant. We augment the data volume by randomly putting noise in TS instances with a random duration. The augmentation degree is set to *10* times of original volume. We put 3 Concept Drifts equally

[2]Especially for TS Classification, hundreds or thousands of TS instances can be considered as BIG DATA, due to the high time complexity of algorithms in the domain which are based on exact distance measure. Our algorithm has a time of $\mathcal{O}(N^2n^3logn)$, N: number of instances, n: TS length.

distributed on time axis. The total labelled instances are sampled into 3 equal-sized subsets with different concepts.

-Reproducibility and Parameters: The initial Shapelet Extraction algorithm is based on [16], where the Shapelet length $m \in [0.1n, 0.5n]$ with a step of $0.25m$, where n is TS length. The advanced *MASS* algorithm (*mass_ v2*) [288] for similarity measure is re-implemented under Python3, where flat subsequences (i.e., those where all values are equal) are ignored by the algorithm, as such a subsequence is meaningless from the definition of Shapelet, and will produce an error during *MASS* computation due to the empty value of its standard deviation. The threshold in Discriminative Profile is replaced by a top-K selection in the profile, which follows the same value as final extracted Shapelet number k ($k = 10$ for each class). As for the loss threshold for Shapelet Evaluation, $\Delta \in \{0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50\}$. For Concept Drift Detection, we take the loss threshold Δ which brings the highest accuracy in raw dataset. The tolerance $\delta \in \{0.10, 0.15, 0.30\}$, PH threshold $\lambda = 0.4$. The TS chunk size is fixed at *5*.

3.5.2 RQ1: Incremental learning with ISMAP

Table 3.1: Shapelet Datasets in UCR Archive used for Incremental Test (ISMAP)

Type	Name	Train/Test	Class	Length	IG	KW	MM	ISMAP(best)	Para. (Δ)	Comp. Ratio
Simulated	SyntheticControl	300/300	6	60	**0.9433**	0.9000	0.8133	0.7007	0.35	46.7%
Sensor	Trace	100/100	4	275	0.9800	0.9400	0.9200	**1**	0.5, 0.45	26.0%
	MoteStrain	20/1252	2	84	0.8251	0.8395	0.8395	**0.9169**	0.45	60.0%
	SonyAIBO.I	20/601	2	70	0.8453	0.7281	0.7521	**0.9151**	0.4	95.0%
	SonyAIBO.II	27/953	2	65	0.8457	-	-	**0.8583**	0.4	63.0%
	ItalyPower.	67/1029	2	24	0.8921	0.9096	0.8678	**0.9466**	0.45	25.4%
ECG	ECG5000	500/4500	5	140	0.7852	-	-	**0.9109**	0.4	9.4%
	ECGFiveDays	23/861	2	136	0.7747	0.8721	0.8432	**0.9826**	0.4	51.2%
	TwoLeadECG	23/1189	2	82	0.8507	0.7538	7657	**0.9337**	0.5	47.8%
Images	Symbols	25/995	6	398	0.7799	0.5568	0.5799	**0.8113**	0.35	96.0%
	Coffee	28/28	2	286	**0.9643**	0.8571	0.8671	0.9286	0.4	78.6%
	FaceFour	24/88	4	350	0.8409	0.4432	0.4205	**0.9886**	except 0.45	62.5%
	DiatomSize.	16/306	4	345	0.7222	0.6111	0.4608	**0.8758**	0.5	50.0%
Motion	GunPoint	50/150	2	150	0.8933	0.9400	0.9000	**0.9733**	0.45	42.0%

Based on the explainable approach proposed in [16], we test firstly the incremental feature of *ISMAP* by adopting interleaved Test-then-Train strategy with an evaluation procedure. The loss threshold $\Delta \in [0.2, 0.5]$, with a step of 0.05, which controls the sensitivity of system for importing TS instance into Shapelet Extraction process.

Baselines: We focus on the feature itself, that is, to select the best quality Shapelets from data source. Therefore, to test the reliability of *ISMAP*, we take the Shapelet Tree methods as baselines, rather than considering classifiers learned over shapelet-transformed data[97][3]. The Shapelet Tree methods utilize different quality measures to extract the

[3]Nevertheless, the extracted high quality Shapelets can be concatenated with Shapelet Transform methods for a higher accuracy, though it's not our focus here.

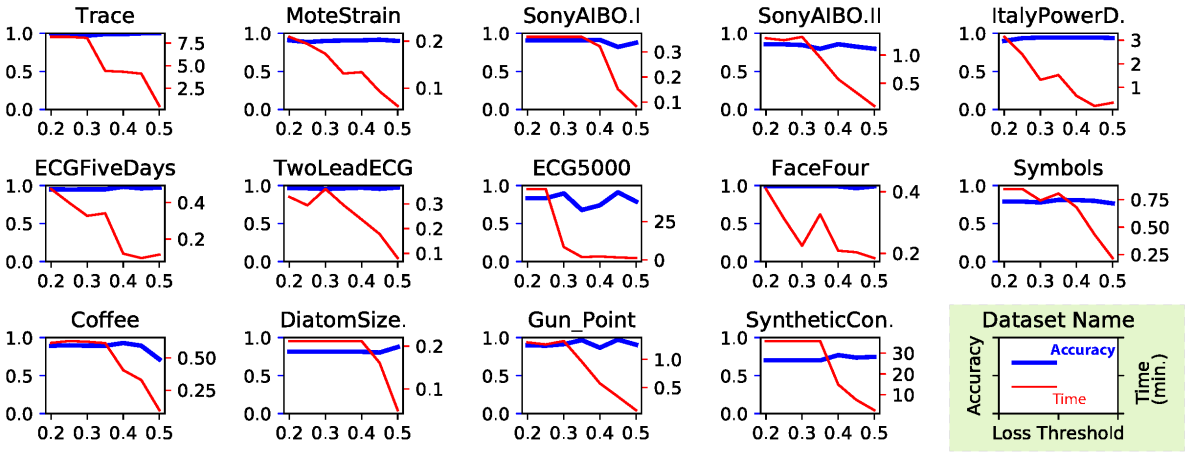


Figure 3.6: Results of Incremental Test (ISMAP) by adopting an extra Shapelet Evaluation procedure

Shapelet and predict target instance: a) Information Gain (IG)[14], b) Kruskall-Wallis (KW) [208], c) Mood's Median (MM) [208]. As quality measure's calculation is negligible compared to the total time cost, the computation time should remain at the same level when they adopt the same distance measure (e.g., *MASS*), and when *ISMAP* doesn't adopt a Test-then-Train strategy.

Table 3.1 shows the accuracy performance comparison between baselines and our approach. Obviously, *ISMAP* achieved the top performance on accuracy metric on more datasets than any other classifier (12 of 14). Specifically for sensor, motion and ECG data, *ISMAP* performs no doubt better than other approaches, and achieved more than 20% accuracy improvement in *ECG5000*.

Besides the accuracy advantage compared to the baselines, the incrementality of *ISMAP* allows a flexible adjustment between accuracy and time cost. **Table** 3.1 shows as well the parameter Δ which brings the best accuracy performance. The parameter sets a loss threshold for Shapelet evaluation during incremental extraction process, and decides whether the input TS instance contains useful information for updating existing Shapelets. The Compression Ratio is defined by the proportion of imported valuable instances over total training instances: $Comp.Ratio = \frac{nbr.instance_{imported}}{nbr.instance_{training}}$, the ratio below 1 brings a better performance in both time and memory cost. **Fig.** 3.6 shows a global view of accuracy and time cost tested by *ISMAP* under different loss thresholds.

In general, we consider that a high loss threshold Δ leads to a high efficiency at the expense of certain accuracy. However, from the result in **Table** 3.1, for most of the datasets, *ISMAP* gets the highest accuracy in range $\Delta \in [0.4, 0.45]$, which is reasonable from **Fig.** 3.3. On the one hand, a threshold loss close to or greater than 0.5 will skip a large number of instances which are around the split point and don't make any quality improvement for current Shapelets. Instead, the instances contain valuable information for adjusting

the quality of current Shapelets and could introduce new candidates. Then classifier will be overfit on small number of instances. On the other hand, the accuracy performance is affected by random input order of TS instances, the first extracted candidate Shapelets will affect the evaluation of later TS instances, which brings a small uncertainty for the number of instances to be imported into the system. For instance, in *FaceFour* dataset, time cost increases when Δ changes from 0.3 to 0.35, that means more instances are imported into system even with the increase of threshold. Which is caused by the uncertainty. A small loss threshold in early stage will be biased towards the initial randomly imported TS instances and extracted Shapelets, and reduce the acceptation space for true discriminative Shapelets coming afterwards.

Nevertheless, most of the time the accuracy keeps on a relative stable stage even with the increase of Δ, which can be explained by the fact that the instances from the same class are highly consistent, and share the common Shapelet features. Therefore, the system efficiency can be largely improved with an exchange of a negligible decrease of accuracy.

3.5.3 RQ2: Adaptive learning with ISMAP

For sake of clarity, we have selected for these experiments datasets where Shapelet-based approach has a strong reliability. Due to space limit, we only show the exploration results for two testing contexts: *ECG5000* dataset in the original space and *Trace* dataset in the augmented space. We focus on the explainable detection process of Concept Drift and the reliability of the adaptive Shapelets around drift areas.

Fig. 3.7 shows the Concept Drift detection process on the two datasets under different loss tolerance level. For *ECG5000*, two drifts were put at time tick *167* and *333*. The Concept Drifts were detected by the system within the time periods [170, 195] and [340, 390], which are in strong accordance with the true drifts in the dataset. The extracted Shapelets before each drift period contain overall information of previous subset. During the drift periods, TS chunks are evaluated to update the current learned concept. A small adjustment time period (i.e., 25 and 50 for two drifts respectively) over the entire subset size proved the strong adaptability of the system. For augmented *Trace* dataset, two drifts were put at time tick *333* and *667*. The drift detection mechanism stays reliable under different loss tolerances, which lead to different time periods for adjusting the learned concept. A high tolerance is capable of relieving the effect of outliers or excessive feedback, and allows only a continuous high loss to be considered as a Concept Drift. Therefore, less chunks are imported into the system which leads to less time cost. From the memory and time plot on right **Fig.** 3.7, at the end of each drift adjustment area, the cached information is largely eliminated, finally only 65 of 500 instances of *ECG5000* dataset, 100 of 1000 ($\delta = 0.15$) or 50 of 1000 ($\delta = 0.30$) instances of *Aug. Trace* dataset, are cached in the memory. The proactive caching elimination mechanism shows its elastic feature. Besides, the later imported chunks requires usually longer calculation time (the step becomes longer in the time plot), as more chunks have been cached.

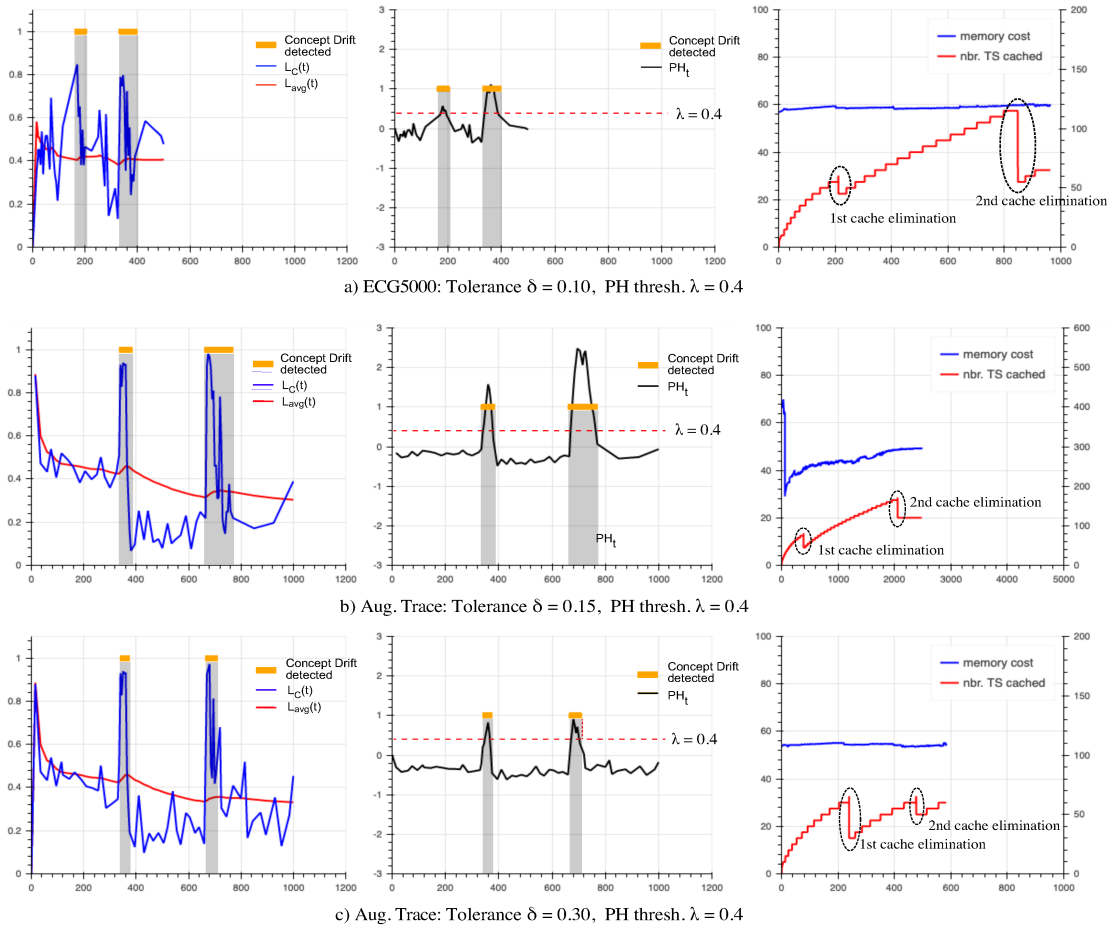


Figure 3.7: Results of Concept Drift Detection on *ECG5000* dataset and augmented *Trace* dataset

In **Table** 3.2, we show the reliability of the Extracted Shapelets on 4 time ticks at the beginning/end of each drift area[4]. The extracted Shapelets perform the same accuracy at the two middle time ticks, which can be explained by the fact that no chunks were imported since the learned *Concept 2* was deemed enough reliable. Globally, the adaptive Shapelets show a high accuracy in such a streaming context with Concept Drift, although the accuracy is a little lower than that in **Table** 3.1, as the training on the subsets gets less information than that on the entire dataset.

3.6 Conclusion

In this chapter, we studied the dynamic feature exploration over Time Series Stream, which is based on the interpretable Shapelet features and an explainable Shapelet extraction pro-

[4]We recall that the *Trace* testing dataset were augmented in the same manner, where drifts were manually added.

Table 3.2: Evaluation in datasets with manually added drift

Dataset	-	**i**(Con. 1)	**ii**(Con. 2)	**iii**(Con. 2)	**iv**(Con. 3)
Aug.Trace	Time tick	*345*	*380*	*670*	*790*
($\delta = 0.15$)	Test Accu.	0.9600	0.9900	0.9900	0.9800
Aug.Trace	Time tick	*350*	*365*	*675*	*700*
($\delta = 0.30$)	Test Accu.	0.9600	0.9800	0.9800	0.9700
*ECG*5000	Time tick	*170*	*195*	*340*	*390*
($\delta = 0.10$)	Test Accu.	0.9018	0.8783	0.8783	0.8927

cess. Multiple temporal relationships between time series instances are considered, which can be either under a stable concept or with a concept drift. First, an incremental Shapelet extraction under stable concept with a novel Shapelet evaluation process is proposed, which improved largely the system's efficiency with an exchange of a negligible decrease of accuracy. Then, for a non-stable concept data source, we adjust the conventional strategies of Concept Drift detection into the context of Time Series Stream, which opens the path for a proactive elimination of data cached in the memory. The system can be applied in the scenario where an existing dataset should be enriched with new knowledge but without human loop in the middle.

As future work, we aim at exploring more challenging scenarios where TS instances are weakly labelled. In addition, we will consider the TS data with more complex structures, such as the multivariate time series (MTS). We continued this research by considering both the complex MTS data and the label constraint in real-life scenarios. However, due to the limitations of the extraction-based Shapelet, such as the high computational cost, we will switch to another TS representation in next chapter for learning MTS with label constraint issues.

Chapter 4

Semi-supervised Learning on Multivariate Time Series

Contents

4.1 Introduction

Most Multivariate Time Series (MTS) data, such as sensor readings, are labeled during the data collection process. The post-labeling of MTS is much more costly than classic data (e.g., image, text, etc.) due to the low interpretability over the real-valued sequence, leading to a considerable constraint for MTS classification in real-life scenarios.

Weakly supervised learning becomes an alternative option for the fully supervised algorithm by learning from the unlabeled samples. The previous studies on weak-label Time Series (TS) learning are usually based on self-training [173] or Positive Unlabeled Learning [175] with a carefully designed distance measure [174] or stopping criterion [176] to learn the pseudo-labels. Besides, they mostly focus on the Univariate Time Series (UTS) with the One-Nearest-Neighbor (1NN) classifier on raw data space, which is widely outpaced by today's advanced approaches [116], such as Deep Neural Networks (DNNs) [119] or ensemble methods [289].

From Univariate Time Series (UTS) to Multivariate Time Series (MTS), traditional methods usually combine the compact and effective features from different variables, such as Shapelet Ensemble [223, 102, 98], global discriminative patterns [290], or bag-of-patterns [224, 225]. However, the predefined features usually fail to capture MTS essentials: the temporal dependency and the interactions of multiple variables (i.e., *spatial interactions*, we use the term *spatial* in this chapter for the variable axis). Recently, some deep learning-based methods were proposed to capture the MTS features with various network structures [230, 226, 21, 291], showing promising performance on MTS classification tasks. However, the above-mentioned methods are mostly fully supervised, and rarely consider the label shortage issue when building the MTS classifier.

The recent research turns to Representation Learning [292] when handling weakly labeled MTS, which allows learning low dimensional embeddings in an unsupervised manner, such as using triplet loss [17] to form the embedding space, then even an SVM classifier is powerful enough on the learned representation [18]. However, existing techniques suffer from three major issues. First, the interactions between the MTS variables are generally computed on the entire 1-D series, ignoring the fact that the local spatial interactions at the sub-sequence level may evolve in the dynamic sequence, that is *spatial dynamics*. Second, the representation learned in a pure unsupervised manner depends mostly on the loss function selection. As no label information is utilized to learn the representation [17], there is a risk that it deviates from the true features, thus affecting the classifier performance. Third, they rather employ deep learning as a blind box and do not focus on the interpretability of the learned representation.

Therefore, to handle both the MTS complex structure and the label shortage prob-

lem, we propose SMATE, **S**emi-supervised Spatio-temporal representation learning on **M**ultiv**A**riate **T**ime S**E**ries. The auto-encoder based structure allows mapping the MTS samples from raw features space $\mathcal{X}$ to low dimensional embedding space $\mathcal{H}$. A Spatial Modeling Block combined with a convolutional network captures the spatial dynamics, whereas a GRU-based structure extracts the temporal dynamic features. Thereby, SMATE is capable of compressing the essential Spatio-temporal characteristics of MTS samples into low-dimensional embeddings. On top of this embedding space $\mathcal{H}$, we propose a semi-supervised three-step regularization process to bring the model to learn class-separable representations, where both the labeled and unlabeled samples contribute to the model's optimization. This regularization process comes with the capability of visualization at each step, making SMATE interpretable.

We summarize our main contributions as follows:

- **Spatio-temporal dynamic features in MTS:** We claim and demonstrate that the temporal dependency and the evolution of the spatial interactions (*spatial dynamics)* are important for building a reliable MTS embedding.
- **Weak supervision on learning representations:** With limited labeled data, SMATE can learn reliable class-separable MTS representations for downstream tasks, such as MTS classification (MTSC).
- **Interpretable MTS embedding learning:** SMATE allows for visual interpretability, not only from the class-separable representations but also in each step of the semi-supervised regularization process.
- **Extensive experiments on the MTS datasets:** The experiments are carried out on various MTS datasets from different application domains. The detailed evaluation with 13 supervised baselines and four semi-supervised work is provided, which shows the effectiveness and the efficiency of SMATE over the state of the art.

The rest of this chapter starts with a review of the most related work. Then, we formulate the research problems. Later, we present in detail our proposal SMATE, which is followed by the experiments on real-life datasets and the conclusion.

4.2 State-of-the-art

We begin by discussing the related work on learning MTS representation with the main extension to MTS Classification (MTSC) tasks. Then, we briefly review the previous work on semi-supervised Time Series learning. Table 4.1 shows the comparison of the methods for learning MTS representation.

4.2.1 Multivariate Time Series Representation Learning

Firstly, we give two primary definitions.

Definition 4.1. (univariate time series). A univariate time series $\mathbf{x} \in \mathcal{R}^T$ is a sequence of real-valued numbers: $\mathbf{x}$=(x_1, x_2, ..., x_i, ..., x_T), where T is the sequence length.

Definition 4.2. (multivariate time series). A multivariate time series $\mathbf{x} \in \mathcal{R}^{T \times M}$ is a sequence of real-valued vectors: $\mathbf{x}$=(x_1, x_2, ..., x_i, ..., x_T), where $x_i \in \mathcal{R}^M$, M is the variable numbers.

A natural way to learn MTS representation is to extend the representation approaches developed earlier on Univariate Time Series (UTS) [31]. This is the case in [222] where the authors further explored Singular Value Decomposition (SVD) with multi-view learning to find the consistency and interactions between variables. Similarly, [223][102] combine Shapelet representation from different variables to build an ensemble-like learner. Symbolic Representation for Multivariate Time Series (SMTS) [224] adopts the Bag-of-Patterns concept, considering all variables simultaneously and constructs a code-book to model the variable relations. Finally, WEASEL+MUSE [225] extend WEASEL [219] from UTS to MTS by creating a histogram of feature counts to capture the local and global changes in relationships between variables.

Different from the interpretable feature-based representations, various deep learning models are proposed to capture the variable (*i.e., spatial*) interactions in MTS. Multi-Channels Deep Convolutional Neural Networks (MC-DCNN) [226] extract firstly 1D-CNN features from each variable, then combine them with a Fully Connected (FC) Layer. Whereas the authors in [227] abandon the combination strategy but apply directly 1D-CNN to all variables. The 2D-CNN features with the cross-attention mechanism in CA-SFCN [125] enhanced the dependencies captured by 1D-CNN on both temporal and spatial axes. Besides, the recurrent models are widely applied to sequential data. A modified Gated Recurrent Unit (GRU) described in [21] models MTS with missing values, where each multivariate step is memorized into state units, then the recurrent structure captures the temporal dependencies. Another group of works [228, 229] adopt Graph Neural Networks (GNNs) to model the spatial interactions. However, they are generally designed for forecasting tasks (e.g., traffic forecasting), and most rely on external information (e.g., the road networks) between the variables. Last but not least, the hybrid LSTM-CNN structure is capable of extracting both local and long-term features. Various work such as the Squeeze-and-Excitation block in MLSTM-FCN [230] or the multi-view learning-like module in TapNet [187], enhanced the hybrid structure via modeling the spatial interactions. However, the interactions are generally computed at the sequence level, ignoring the fact that the local spatial interactions at the sub-sequence level may evolve in the dynamic sequence, i.e., *spatial dynamics*. Moreover, they are all fully supervised, requiring huge labels during the training process. Also, the learned representations are result-oriented (e.g., pursuing higher accuracy), with less focus on the interpretability, considered by the data mining community.

Table 4.1: Existing methods for learning MTS Representation

	SMTS	MUSE	Shapelet	USRL	TapNet	MLSTM-FCN	CA-SFCN	1NN-DTW	**SMATE**
Temporal Dynamics	-	-	-	✓	✓	✓	✓	-	✓
Spatial Dynamics	-	-	-	-	-	-	✓	-	✓
Interpretable Representation	✓	✓	✓	-	✓	-	-	-	✓
Label Shortage	-	-	-	✓	✓	-	-	✓	✓

4.2.2 Semi-supervised Learning on Time Series

The pioneering work [173, 174] on Semi-supervised TS Learning are based on self-training or Positive Unlabeled Learning [175] with the Nearest-Neighbor (1NN) classifier and a carefully designed distance, such as DTW [173] or DTW-D [174], and optimized stopping criterion [176] for importing the pseudo-labels. Those work are followed by [177, 178] for wider contexts. Though not mentioned in their papers, the self-training framework is extensible from the UTS to the MTS setting by using an adapted distance, such as DTW_I [179], DTW_D [179] or DTW_A [180]. However, under more complex scenarios nowadays, such as 30 UEA MTS datasets [182] collected from different domains, the distance-based classifiers show limited performance and are rather used as baselines by recent studies [116].

Learning meaningful MTS representations [292] in a weakly supervised setting draws much attention recently. Unsupervised Scalable Representation Learning (USRL) described in [17] combines causal dilated convolutions with triplet loss for contrastive learning. On the one hand, it learns better UTS representation than the traditional supervised CNNs [119]. On the other hand, a single SVM on the learned MTS representation offers higher accuracy than a DTW_D-based classifier [179]. Similarly, authors in [178] adopt Multi-Task Learning (MTL) to learn the self-supervised UTS features from an auxiliary forecasting task. The recent work Semi-TapNet [187] proposes an Attentional Prototype Network to learn from the unlabeled samples. However, the above-mentioned approaches do not explore thoroughly both the labeled samples and the *Spatio-temporal dynamics* in MTS.

4.3 Problem Formulation

In this section, we formulate firstly the *Spatio-temporal dynamics* learning problem in MTS. Then, we give a formal definition of a semi-supervised classification problem for MTS. Table 4.2 summarizes the notations used in the chapter.

4.3.1 Spatio-temporal Representation for MTS

Table 4.2: Notation

Notation	Description
$\mathcal{D}, \mathcal{D}_l, \mathcal{D}_u$	dataset, labeled portion, unlabeled portion
T, M, N	MTS length, variable numbers, dataset size
L, D	embedding length, embedding dimension size
$m, \mathcal{P}$	temporal window size, embedding pool size
$\mathbf{x}$, $\mathbf{h}$	MTS instance, latent embedding
$\mathbf{s}$	variable/spatial interaction
θ	general parameters to be optimized

The Spatio-temporal modeling of MTS requires considering both the temporal dependency $p(x_{t'}|x_t)$ $(t' > t)$ and the spatial interactions between the variables. Previous studies [187, 230] usually consider the spatial interactions at the sequence level: $\mathbf{s} = \{\mathbf{x}^i \bowtie \mathbf{x}^j\} \in \mathcal{R}^M$, where $\mathbf{x}^i, \mathbf{x}^j \in \mathcal{R}^{T\times 1}$, $\bowtie$ indicates the interactions between the variables. However, the local spatial interactions at the subsequence level $s_t = \{\mathbf{x}^i_{t-m/2,t+m/2} \bowtie \mathbf{x}^j_{t-m/2,t+m/2}\} \in \mathcal{R}^M$ may evolve in the dynamic sequence, where m is the window size. To illustrate, given the system status at time t in MTS, it is not only decided by the local value $x_t \in \mathcal{R}^M$ given a temporal status, but also by its neighbor values $\left[x_{t-m/2} : x_{t+m/2}\right] \in \mathcal{R}^{M\times m}$, which brings a spatial correlation matrix on temporal neighbors and spatial variables given a spatial status $s_t \in \mathcal{R}^M$. Unlike the work such as DCRNN [229] relying on external information (e.g., the road networks) to model the spatial dependency between the variables, we aim at exploring the inherent spatial/variable interactions only on the MTS data.

Therefore, given a sample $\mathbf{x} \in \mathcal{R}^{T\times M}$ in raw space $\mathcal{X}$, the Spatio-temporal representation learning for MTS is to learn a low-dimensional representation $\mathbf{h} \in \mathcal{R}^{L\times D}$ on embedding space $\mathcal{H}$, integrating both temporal dynamic $p(x_{t'}|x_t)$ and spatial dynamic $p(s_{t'}|s_t)$. The item *dynamic* refers to the unstable system status within the evolving multivariate sequential data.

4.3.2 Semi-Supervised Learning on MTS

Definition 4.3. (weak-label MTS dataset). A weakly labeled MTS dataset $\mathcal{D} = \{\mathcal{D}_l, \mathcal{D}_u\}$ contains both labeled and unlabeled MTS samples:

$$\mathcal{D}_l = \{\mathbf{x}_i, y_i\}_{i=1}^{N*r},\ \mathcal{D}_u = \{\hat{\mathbf{x}}_i\}_{i=1}^{N*(1-r)}$$

where r $(0 \le r \le 1)$ indicates the *ratio* of the labeled samples in $\mathcal{D}$ of size N, y_i is the annotation of the labeled instance $\mathbf{x}_i$.

The semi-supervised MTS learning aims at training a classifier to predict successfully the label of a testing MTS sample, adopting the supervised training from $\mathcal{D}_l$ and further unsupervised adjustment/optimization from $\mathcal{D}_u$.

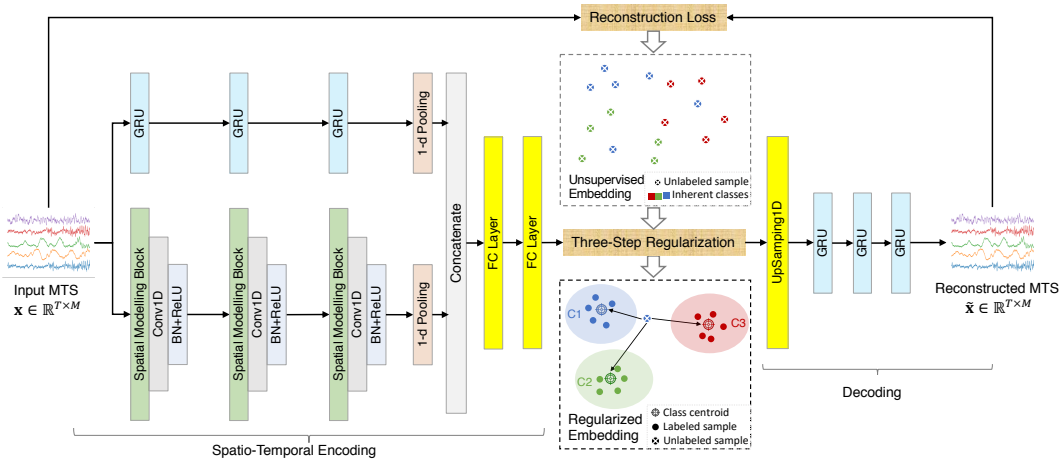


Figure 4.1: Model Structure of SMATE

4.4 Proposal: SMATE

In this section, we introduce SMATE, which captures the essential characteristics of Multivariate Time Series (MTS) and allows learning an interpretable representation space in a semi-supervised manner. This section is organized as follows. First, we introduce the global model structure of SMATE. Then, we describe how the Spatio-temporal representation is learned from the raw MTS data space. Finally, we give the joint model optimization, which coordinates the weak supervision and the embedding learned via a three-step regularization process.

4.4.1 Global Structure of SMATE

SMATE is based on an *asymmetric auto-encoder* structure, integrating three key components: Spatio-temporal dynamic encoder, sequential decoder, and semi-supervised three-step regularization of the embedding space.

Given the fact that extracting features from a high-dimensional space generally requires additional attention compared to restoring data from a low-dimensional space [292], the encoding and decoding process in SMATE adopts different weight matrix (i.e., *asymmetric auto-encoder*) to capture the inner structure of MTS data. Although recent work [293] does represent the temporal dynamics of MTS via a sequence to sequence (seq2seq) model, they do not encompass the complex Spatio-temporal structure of MTS. As shown in Figure 4.1, SMATE adopts a two-channel encoder exploring both spatial and temporal dynamics and embeds the input MTS into a low-dimensional representation space, where the embedded samples are sparsely distributed with the reconstruction-based optimization. On the unsupervised embedding space, a three-step regularization is applied to learn class-separable embeddings. The class centroids are regularized by labeled and unlabeled samples, show-

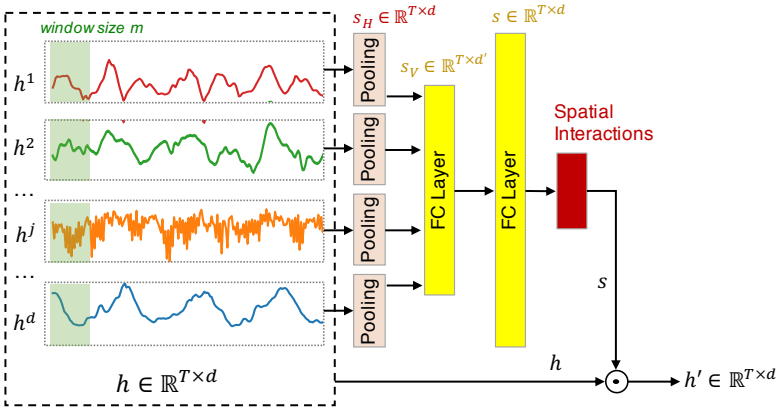


Figure 4.2: The Spatial Modeling Block (SMB)

ing interpretability over the representation space. Finally, the model is jointly optimized by the reconstruction objective and the regularization objective.

4.4.2 Spatial Modeling Block (SMB)

Firstly, we introduce a novel module, Spatial Modeling Block (SMB), to capture the spatial interactions at subsequence levels. As shown in Figure 4.2, SMB takes as input an MTS representation $h = \{h_i\}_{i=1}^{T} \in \mathcal{R}^{T\times d}$ ($d = M$ for the first block in spatial encoding channel), followed by a one-dimensional average pooling layer on each variable $h^j \in \mathcal{R}^{T\times 1}$, encoding the temporal neighbors into the horizontal system status $s_H(i)$=$avg([h_{i-m/2} : h_{i+m/2}])$, where i is the time tick, m is the window size. Then the Fully Connected (FC) layers allow firstly the interaction of the horizontal system status s_H in the vertical direction via a low-dimensional compression $s_V \in \mathcal{R}^{T\times d'}$, then remapping it to the initial data space to decide the spatial interaction weights at each one-dimensional segment. We define the spatial interactions $\mathbf{s} = \{s_i\}_{i=1}^{T}$, where $s_i \in \mathcal{R}^d$, representing the interaction weights for the vector $h_i \in \mathcal{R}^d$. The output of SMB is described by h'=$h \odot \mathbf{s}$, with the calibrated weights for each 1-D TS segment, where $\odot$ is the element-wise multiplication.

4.4.3 Spatio-Temporal Encoding on MTS

Given $\mathbf{x} \in \mathcal{R}^{T\times M}$, the low-dimensional representation $\mathbf{h} \in \mathcal{R}^{L\times D}$ embeds the Spatio-temporal features of $\mathbf{x}$ by a neural network-based function $f_\theta(\mathbf{x})$. The low-dimensional embedding brings dramatic improvement on both the efficiency and accuracy for classification tasks [187], owning to the fact that the classifier is not distracted by the redundant information in raw data.

As shown in Figure 4.1, we adopt a two-channel structure to encode the spatial and temporal features in MTS, respectively. For the temporal channel, among different variants

of the recurrent neural networks (RNN), we specifically consider Gated Recurrent Units (GRUs) [21] that mitigate the vanishing gradient problem. An update gate z_t and a reset gate r_t control the hidden state $h_t \in \mathcal{R}^{d_g}$ with the observation $x_t \in \mathcal{R}^M$ and the previous hidden state $h_{t-1} \in \mathcal{R}^{d_g}$, where d_g is the output dimension of GRUs. The update functions are as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{4.1}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{4.2}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h(h_{t-1} \odot r_t) + b_h) \tag{4.3}$$

where W_x, U_x, b_x $(x \in [r, z, h])$ are model parameters, σ is the sigmoid function, $\odot$ is the element-wise multiplication. Three GRUs are cascaded with a 1-D pooling layer to output $\mathbf{h}(\mathbf{T}) \in \mathcal{R}^{L \times d_g}$, where $L{=}T/\mathcal{P}$, $\mathcal{P}$ is the pool/sampling size[1].

For the spatial channel, we define the convolutional module:

$$\mathbf{h}'(l) = SMB(\mathbf{h}(l)) \tag{4.4}$$

$$\mathbf{h}(l+1) = ReLU(BN(W \otimes \mathbf{h}'(l) + b) \tag{4.5}$$

where l $(0 \leq l < 3)$ is the module number, $\mathbf{h}(0){=}\mathbf{x} \in \mathcal{R}^{T \times M}$, $\mathbf{h}(l) \in \mathcal{R}^{T \times d_c}$, d_c is the filter number, W is the 1-D convolutional kernel of size m, $\otimes$ is the convolution operator. Within each of the three modules, the SMB firstly calibrates the interaction weights for each 1-D segment and outputs $\mathbf{h}' \in \mathcal{R}^{T \times d}$. Then a 1-D convolutional layer concatenated with Batch Normalization [294] and Rectified Linear Units (ReLU) [295] is deployed. The 1-D kernels match with the window size m in their neighboring SMB, as the convolution product $W \otimes \left[h'_{i-m/2} : h'_{i+m/2}\right]$ requires considering the spatial interactions captured by SMB within the same interval. Similar to the temporal channel, a 1-D pooling layer is applied after the last convolutional block to output $\mathbf{h}(\mathbf{S}) \in \mathcal{R}^{L \times d_c}$. Finally, we output the combined spatial and temporal features $h_{concat} = concat(\mathbf{h}(\mathbf{T}), \mathbf{h}(\mathbf{S})) \in \mathcal{R}^{L \times (d_g + d_c)}$ and apply two FC layers to get the Spatio-temporal embedding $\mathbf{h} \in \mathcal{R}^{L \times \mathbf{D}}$. The matrix representation allows the maximal preservation of the Spatio-temporal features and facilitating the MTS restoration. The detailed parameter settings can be found in Section 4.5.1.2.

4.4.4 Joint Model Optimization

As shown in Figure 4.1, since the representation learned via an autoencoder-based structure generally has a sparse distribution of class-specific samples [292], the unsupervised training derived from the reconstruction objective does not consider thoroughly the inner relation between class-specific samples but focus on the restoration performance from the embeddings. To address the issue, we propose a joint model optimization that integrates

[1]Note that the pool/window size m in SMB and $\mathcal{P}$ are different parameters.

the temporal reconstruction and the three-step regularization objectives. Specifically, the joint optimization combines both the labeled and unlabeled samples to learn the class-specific clusters on the embedding space.

Firstly, we define the *temporal reconstruction loss* as:

$$L_R = \sum\nolimits_t \|x_t - \tilde{x}_t\|_2 \tag{4.6}$$

where $x_t, \tilde{x}_t \in \mathcal{R}^M$, corresponding to the observations in the raw and reconstructed MTS instances $\mathbf{x}$ and $\tilde{\mathbf{x}}$.

Then, we introduce the three-step regularization combining both labeled and unlabeled samples to foster the model adaptation of class-separable embeddings. The regularization approaches the embeddings within the class-specific clusters to the virtual class centroids, which are trained progressively.

***Step 1* -Supervised Centroids Initialization**: The class centroids are initialized by the class-specific embeddings. Given the labeled training set $\mathcal{D}_l = \{X^k\}_{k=1}^K$ where K is the class number, $X^k \in \mathcal{R}^{N_k \times T \times M}$ is a sample collection of class k, N_k is the sample number of class k. Then the embedding set $H^k{=}f_\theta(X^k) \in \mathcal{R}^{N_k \times L \times D}$ initializes the class centroid $\mathbf{c}_k$ by:

$$\mathbf{c}_k = mean(H^k), \quad \mathbf{c}_k \in \mathcal{R}^{L \times D} \tag{4.7}$$

***Step 2* -Supervised Centroids Adjustment**: Once the centroids are initialized, we can make the supervised adjustment since the distance-based class probability allows to assess the contribution of individual samples on the centroid's decision. In other words, the centroid $\mathbf{c}_k$ is affected by larger contribution weights brought by nearby samples of class k. Let $\mathbf{x}_i^k \in \mathcal{R}^{T \times M}$ be a time series of class k, we define the weight of $\mathbf{x}_i^k$ to $\mathbf{c}_k$ as the inverse Euclidean Distance (ED) between the embedding $\mathbf{h}_i^k = f_\theta(\mathbf{x}_i^k) \in \mathcal{R}^{L \times D}$ and the centroid $\mathbf{c}_k$:

$$W_{k,i} = 1 - \frac{ED(\mathbf{h}_i^k, \mathbf{c}_k)}{\sum_{j=1}^K ED(\mathbf{h}_i^k, \mathbf{c}_j)} \tag{4.8}$$

Then the class centroid $\mathbf{c}_k$ can be adjusted accordingly by the labeled samples within the class-specific cluster:

$$\mathbf{c}_k = \sum\nolimits_{i=1}^{N_k} W_{k,i} \cdot \mathbf{h}_i^k, \quad \mathbf{h}_i^k \in H^k \tag{4.9}$$

***Step 3* -Unsupervised Centroids Adjustment**: Given the unlabeled samples $\mathcal{D}_u = \{\hat{\mathbf{x}}_i\}_{i=1}^{N*(1-r)}$, where r is the labeled data ratio. Apart from the optimization from the reconstruction objective, the unlabeled sample $\hat{\mathbf{x}}_i$ is capable of adjusting the centroid c_k via the propagated label from the distance-based class probability defined as:

$$\hat{p}_\theta(y = k|\hat{\mathbf{x}}_i) = 1 - \frac{ED(f_\theta(\hat{\mathbf{x}}_i), \mathbf{c}_k)}{\sum_{j=1}^K ED(f_\theta(\hat{\mathbf{x}}_i), \mathbf{c}_j)} \tag{4.10}$$

The unlabeled sample $\hat{\mathbf{x}}_i$ will be then integrated into the class-specific cluster with the highest probability. We can further adjust the class centroid $\mathbf{c}_k$ considering the unlabeled samples:

$$\mathbf{c}_k = \frac{N_k}{N_k + \hat{N}_k} \sum_{i=1}^{N_k} W_{k,i} \cdot \mathbf{h}_i^k + \frac{\hat{N}_k}{N_k + \hat{N}_k} \sum_{i=1}^{\hat{N}_k} \hat{p}_{k,i} \cdot \hat{\mathbf{h}}_i^k \tag{4.11}$$

where $\hat{\mathbf{h}}_i^k = f_\theta(\hat{\mathbf{x}}_i^k)$, $\hat{N}_k$ is the number of samples of class k in $\mathcal{D}_u$ with the propagated label.

The class centroids are initialized and adjusted by both labeled and unlabeled samples on the embedding space, from which we formalize the *regularization loss* derived from the labeled samples as follows:

$$L_{Reg}(\theta) = -\sum\nolimits_k log W_\theta(y = k|\mathbf{x}) \tag{4.12}$$

As both the reconstruction and regularization losses are normalized, we define the global optimization objective as:

$$\min_\theta(L_R + \lambda L_{Reg}) \tag{4.13}$$

where $\lambda \geq 0$ is a hyperparameter that balances the two losses. Importantly, L_{Reg} is included such that the embedding process not only serves to reduce the dimensions – it is actively conditioned to facilitate the encoder in learning class-separable embeddings. In practice, SMATE is not sensitive to λ (see Fig. 4.3 in Section 4.5.2.3); then for all the experiments, we set $\lambda = 1$.

4.5 Experiments

In this section, we evaluate the performance of the Spatio-temporal representation learned by SMATE. Firstly, we show the experimental setup, including the dataset information, hyperparameters' setting, baseline descriptions, and evaluation metrics. Then we evaluate the performance of the model with different baselines on both supervised and semi-supervised learning tasks. Finally, we analyze the Spatial Modeling Block regarding its ability to model the dimensional interactions in MTS. The model was trained using the Adam optimizer [296] on a single Tesla V100 GPU of 32 Go memory with CUDA 10.2. The authors are devoted to promoting reproducibility. Therefore, the source code, datasets, and instructions are publicly available[2].

[2]https://github.com/SMATE2021/SMATE

4.5.1 Experimental setup

We evaluate the learned MTS representation on both classification and semi-supervised classification tasks. As SMATE allows learning class-separable representations, then an SVM classifier is powerful enough [18] to validate the learned representations. For the classification task, we firstly adopt the full labeled training set to learn the class-separable representations, then we train an SVM classifier with radial basis function kernel on the embedding space. For the semi-supervised aspect, we apply different portions of training labels to train the semi-supervised representation model, serving to learn the SVM classifier with the propagated labels from the distance-based class probability.

In brief, the experiments were designed to answer the following research questions (RQs):

RQ 1 *Classification Performance.* How well our SMATE model performs on the classification tasks when adopting the complete annotations?

RQ 2 *Semi-supervised Classification Performance.* How successful is SMATE in semi-supervised learning when adopting part of labels for learning the representation?

RQ 3 *Interpretation over the Representation Space.* How the learned representation space can be interpreted?

RQ 4 *Performance of Spatial Modeling Block (SMB).* How well the proposed SMB module performs for modeling the variable interactions?

RQ 5 *Model Efficiency.* How efficient is SMATE compared to the previous models?

4.5.1.1 Datasets description

We evaluate our proposed method on the newly released UEA archive [182], including 30 MTS datasets from various application domains[3], which have a big difference in dimension size ($2 \sim 963$), sample length ($8 \sim 3000$) and the number of training samples ($12 \sim 7494$). Readers can find information about the selected datasets sin Table 4.3. We adopt the default train/test split of the archive. All 30 datasets are chosen for supervised analysis, whereas the datasets {*ArticularyWordR., Epilepsy, Heartbeat, SelfRegulationSCP1*} from four different domains are adopted for semi-supervised study.

4.5.1.2 Hyperparameters Setting

[Network Architecture] As shown in Table 4.4, we set 3 GRU layers with a hidden dimension size of 128 for the Temporal Channel. Two 1-D Average Pooling layers are applied for Temporal and Spatial Channels, respectively, where we give the pool size,

[3]The datasets can be found in www.timeseriesclassification.com

Table 4.3: MTS dataset information

Domain	Dataset	Samples	Dim. (M)	Length (T)	Class
Human Activity	BasicMotions	40/40	6	100	4
	Cricket	108/72	6	1197	12
	Epilepsy	137/138	3	206	4
	ERing	30/270	4	65	6
	Handwriting	150/850	3	152	26
	Libras	180/180	2	45	15
	N/ATOPS	180/180	24	51	6
	RacketSports	151/152	152	30	4
	UWaveGestureLibrary	120/320	6	30	4
Motion	ArticularyWordR.	275/300	9	144	25
	CharacterTrajectories	1422/1436	3	182	20
	EigenWorms	128/131	6	17984	5
	PenDigits	7494/3498	2	8	10
ECG	AtrialFibrillation	15/15	2	640	3
	StandWalkJump	12/15	4	2500	3
EEG/ MEG	FaceDetection	5890/3524	144	62	2
	FingerMovements	316/100	28	50	2
	HandMovementDirection	160/74	10	400	4
	InsectWingbeat	30000/20000	200	30	10
	JapaneseVowels	270/270	12	29	9
	MotorImagery	278/100	64	3000	2
	SelfRegulationSCP1	268/293	6	896	2
	SelfRegulationSCP2	200/180	7	1152	2
Audio Spectra	DuckDuckGeese	50/50	1345	270	5
	Heartbeat	204/205	61	405	2
	Phoneme	3315/3353	11	217	39
	SpokenArabicDigits	6599/2199	13	93	10
Others	EthanolConcent.	261/263	3	1751	4
	LSST	2459/2466	6	36	14
	PEMS-SF	267/173	963	144	7

Table 4.4: Network Architecture of SMATE

Module	Layer	Type
Temporal Channel	1	GRU (128)
	2	GRU (128)
	3	GRU (128)
	4	AveragePooling1D($\mathcal{P}$, $\mathcal{P}$, 0)
Spatial Channel	1	SMB_1 + Conv1D(8,1,0) -128 filters + Batch Norm + ReLU
	2	SMB_2 + Conv1D(5,1,0) -256 filters + Batch Norm + ReLU
	3	SMB_3 + Conv1D(3,1,0) -128 filters + Batch Norm + ReLU
	4	AveragePooling1D($\mathcal{P}$, $\mathcal{P}$, 0)
FC	1	FC (128) + Batch Norm + LeakyReLU
	2	FC (128) + Batch Norm
SMB_1	1	AveragePooling1D(8, 1, 0)
	2	FC ($d^{'}$) + ReLU
	3	FC (M) + Sigmoid
SMB_2	1	AveragePooling1D (5, 1, 0)
	2	FC (8) + ReLU
	3	FC (128) + Sigmoid
SMB_3	1	AveragePooling1D (3, 1, 0)
	2	FC (16) + ReLU
	3	FC (256) + Sigmoid
Decoder	1	UpSampling1D ($\mathcal{P}$)
	2	GRU (128)
	3	GRU (128)
	4	GRU (M)

stride and padding in the bracket. We provide the kernel size, stride, and padding in the brackets of *Conv1D*. The SMBs are configured with consistent parameters of their neighbor *Conv1D*.

However, as the datasets are collected from different domains with a big difference in M, T, and N, it is impractical to apply a unified parameter setting on all datasets. The kernel size m of the three convolution modules is set to (8,5,3), except the *PenDigits* dataset for which the kernels are set to (4,1,1) as it is infeasible to apply our default kernel size "8" into an 8-length time series. The hidden dimension size p' in SMB and pool size $\mathcal{P}$ are set as follows:

$$d' = \begin{cases} M/10, & 100 \leq M \\ M/4, & 10 \leq M < 100 \\ M, & M < 10 \end{cases}, \mathcal{P} = \begin{cases} T/20, & 1000 \leq T \\ T/10, & 50 \leq T < 1000 \\ T/4, & T < 50 \end{cases} \tag{4.14}$$

[Experiment Parameters] The Adam optimizer is set with the learning rate of 0.00001 and the default exponential decay rate in Keras. As there are limited training samples in most of the UEA datasets, it is not feasible to separate a validation set from the small size of training samples. For instance, the dataset "StandWalkJump" contains only 12 training samples for three classes. It is impractical to split the small training samples into training and validation sets. Therefore, for the datasets with less than 100 training samples, we

define the stop condition based on the training loss. For the rest, the validation split is set to 0.2. We set the stop condition to hold when the difference of training/validation loss between epochs is less than a small threshold, 0.0001 for three consecutive steps.

4.5.2 RQ 1: Classification Performance Evaluation

We use the accuracy as the default metric for the supervised tasks, which is the default criterion in Time Series Classification work [116]. We also report the number of Win/Ties and the average rank [187] of different methods.

4.5.2.1 Comparison Methods

We compare the performance of a classification task with 13 benchmark approaches, including both classical data mining and recent deep learning methods. We adopt the default parameter settings described in each paper for testing. The methods are summarized as follows:

- Distance-based Nearest Neighbor (1NN) on non-normalized (*non-norm*) or normalized (*norm*) MTS [179]. **1NN-ED** (*non-norm* & *norm*): Euclidean Distance; **1NN-DTW**$_I$ (*non-norm* & *norm*): Sum of Dynamic Time Warping (DTW) distance [180] on each variable; **1NN-DTW**$_D$ (*non-norm* & *norm*): DTW distance applied directly on multi-variate vectors; **1NN-DTW**$_A$ (*norm*): Adaptive distance selected between DTW$_I$ and DTW$_D$ with higher accuracy at run time.

- Bag-of-patterns classifier. **WEASEL+MUSE**[225]: the logistic regression classifier on top of the bag of discriminative features that are extracted from different variables.

- Deep Learning-based classifier. **USRL** [17]: SVM classifier on the representation learned via unsupervised temporal encoding, the Contrastive Learning on Triplet Loss is adopted for adjusting the representation space; **TapNet** [187]: a Softmax function over MTS embeddings, which are learned from a set of variable combinations (i.e., multi-view on the variables); **MLSTM-FCN** [230]: a multi-layer perceptron (MLP) with Softmax function over the concatenated LSTM and CNN layers, capturing the spatial interactions between 1-D series; **CA-SFCN** [125]: Cross Attention Mechanism on both temporal and spatial axes, working with Fully Convolutional Networks; **SMATE**$_{\mathbf{NR}}$: SMATE without supervised Regularization, instead, a *Softmax* layer is applied on the embedding.

4.5.2.2 Results Analysis

Table 4.5 shows the accuracy results comparison between our proposition and the 13 baselines mentioned above. We show as well the average rank and the number of Wins/Ties

Table 4.5: Performance Comparison for MTS classification over UEA MTS archive

Dataset	SMATE	$SMATE_{NR}$	USRL	TapNet	MLSTM -FCN	CA-SFCN	WEASEL +MUSE	1NN-ED	1NN-DTW_I	1NN-DTW_D	1NN-ED (*norm*)	1NN-DTW_I (*norm*)	1NN-DTW_D (*norm*)	1NN-DTW_A (*norm*)
ArticularyWordR.	**0.993**	0.987	0.987	0.987	0.973	0.97	0.99	0.97	0.98	0.987	0.97	0.98	0.987	0.987
AtrialFibrillation	0.133	0.133	0.133	**0.333**	0.267	**0.333**	**0.333**	0.267	0.267	0.2	0.267	0.267	0.22	0.267
BasicMotions	**1**	**1**	**1**	**1**	0.95	**1**	**1**	0.675	**1**	0.975	0.676	**1**	0.975	**1**
CharacterTrajectories	0.984	**0.997**	0.994	**0.997**	0.985	0.988	0.99	0.964	0.969	0.99	0.964	0.969	0.989	0.989
Cricket	0.972	0.964	0.986	0.958	0.917	0.972	**1**	0.944	0.986	**1**	0.944	0.986	**1**	**1**
DuckDuckGeese	N/A	N/A	**0.675**	0.575	**0.675**	N/A	0.575	0.275	0.55	0.6	0.275	0.55	0.6	0.567
EigenWorms	N/A	N/A	0.878	0.489	0.504	N/A	**0.89**	0.55	0.603	0.618	0.549	N/A	0.619	N/A
Epilepsy	0.964	0.946	0.957	0.971	0.761	0.986	**1**	0.667	0.978	0.964	0.666	0.978	0.964	0.979
ERing	**0.97**	0.904	0.88	0.904	0.941	0.856	0.964	0.93	0.93	0.93	0.93	0.93	0.93	0.93
EthanolConcentration	0.373	0.373	0.236	0.323	0.373	0.323	**0.43**	0.293	0.304	0.323	0.293	N/A	0.323	0.316
FaceDetection	**0.563**	0.545	0.528	0.556	0.545	N/A	0.545	0.519	0.513	0.529	0.519	0.5	0.529	0.529
FingerMovements	**0.59**	0.55	0.54	0.53	0.58	**0.59**	0.49	0.55	0.52	0.53	0.55	0.52	0.53	0.509
HandMovementD.	**0.527**	0.378	0.27	0.378	0.365	0.324	0.365	0.279	0.306	0.231	0.278	0.306	0.231	0.224
Handwriting	0.426	0.335	0.533	0.357	0.286	0.322	0.605	0.371	0.509	**0.607**	0.2	0.316	0.286	0.601
Heartbeat	0.727	0.619	0.737	0.751	0.663	**0.756**	0.727	0.62	0.659	0.717	0.619	0.658	0.717	0.571
InsectWingbeat	N/A	N/A	0.16	**0.208**	0.167	N/A	N/A	0.128	N/A	0.115	0.128	N/A	N/A	N/A
JapaneseVowels	0.978	0.967	**0.989**	0.965	0.976	0.973	0.973	0.924	0.959	0.949	0.924	0.959	0.949	0.959
Libras	0.849	0.834	0.867	0.85	0.856	0.89	0.878	0.833	**0.894**	0.872	0.833	**0.894**	0.87	0.879
LSST	0.591	0.575	0.558	0.568	0.373	**0.674**	0.59	0.456	0.575	0.551	0.456	0.575	0.551	0.551
MotorImagery	**0.59**	**0.59**	0.54	**0.59**	0.51	N/A	0.51	0.39	N/A	0.5	0.51	N/A	0.5	0.5
N/ATOPS	0.883	0.85	0.944	0.939	0.889	**0.956**	0.87	0.86	0.85	0.883	0.85	0.85	0.883	0.883
PEMS-SF	**0.763**	0.751	0.688	0.751	0.699	N/A	N/A	0.705	0.734	0.711	0.705	0.734	0.711	0.73
PenDigits	0.98	0.98	**0.983**	0.98	0.978	0.975	0.948	0.973	0.939	0.977	0.973	0.939	0.977	0.977
Phoneme	0.177	0.19	**0.246**	0.175	0.11	0.19	0.19	0.104	0.151	0.151	0.104	0.151	0.151	0.151
RacketSports	0.829	0.816	0.862	0.868	0.803	0.875	**0.934**	0.868	0.842	0.803	0.868	0.842	0.803	0.858
SelfRegulationSCP1	**0.887**	0.874	0.846	0.739	0.874	0.734	0.71	0.771	0.765	0.775	0.771	0.765	0.775	0.786
SelfRegulationSCP2	**0.556**	0.533	**0.556**	0.55	0.472	N/A	0.46	0.483	0.533	0.539	0.483	0.533	0.539	0.539
SpokenArabicDigits	0.982	0.967	0.956	0.983	**0.99**	0.982	0.982	0.967	0.96	0.963	0.967	0.959	0.963	0.963
StandWalkJump	**0.533**	0.4	0.4	0.4	0.067	0.2	0.333	0.2	0.333	0.2	0.2	0.333	0.2	0.333
UWaveGestureLibrary	0.897	0.869	0.884	0.894	0.891	0.8	**0.916**	0.881	0.868	0.903	0.81	0.868	0.903	0.9
Avg. Rank	3.80	5.93	5.7	4.83	7.33	5.41	4.93	9.07	7.52	6.4	9.5	7.81	6.85	6.25
Wins (Ties)	**11**	**3**	6	5	2	6	8	0	2	2	0	2	1	2

of each method. "N/A" indicates the model is not applicable due to memory overflow. Overall, SMATE defends its reliability with 11 Wins/Ties and the highest average rank of 3.85 among all the baselines. The current state-of-the-art deep learning method (TapNet, CA-SFCN) and the powerful data mining method (WEASEL+MUSE) have close ranks (4.73/5.45/4.66). CA-SFCN performs the best on five datasets but is not applicable on seven datasets due to memory overflow. WEASEL+MUSE performs among the best in Human Activity Recognition tasks (*BasicMotions*, *Criket*, *Epilepsy*), as the class-discriminative patterns can be directly extracted from the raw data space. Besides, the unsupervised representation learning method (USRL) performs much worse than SMATE with the same SVM classifier, confirming the reliability of our supervised regularization on the embedding space. Moreover, SMATE achieves the best performance among the baselines on all the datasets of EEG/MEG applications [182] (*FaceDetection*, *FingerMovements*, *HandMovementDirection*, *MotorImagery*, *SelfRegulationSCP1*, *SelfRegulationSCP2*), where the signals (i.e., variables) generally have strong and dynamic dependencies with each other. The spatial dynamic interactions could be essential characteristics that SMATE has successfully captured. SMATE_{NR} performs much worse than SMATE, proving that the supervised regularization process helps to build the class-separable embeddings.

However, SMATE produces visibly low accuracy on some datasets, e.g., 0.133 on *AtrialFibrillation*, 0.177 on *Phoneme*, on which the baselines perform poorly as well. This is probably caused by the original data source.

4.5.2.3 Parameter effects on model performance

We analyse the effects of window/kernel size m, pool size $\mathcal{P}$ and regularization weight λ on the example dataset *SelfRegulationSCP1*. In Fig. 4.3, we observe that a larger m brings higher model accuracy, as it creates a larger receptive field captured by both SMB and *Conv1D* block. Furthermore, the early convolutional modules are more sensitive to m as they keep close to the raw input features. A larger pool size $\mathcal{P}$ will reduce the embedding size, thus affecting the preserved embedding features and the training efficiency. SMATE is not sensitive to λ. It can be explained by the fact that a stable reconstruction process makes the model focus more on the regularization of the embedding space. When λ=0, no regularization is applied, leading to sparsely distributed embeddings with poor prediction performance.

4.5.3 RQ 2: Semi-supervised Classification Performance

For semi-supervised tasks, we evaluate the classifier's accuracy at different supervision levels by varying the labeled samples in the training set. We select the datasets {*Articulary-WordR., Epilepsy, Heartbeat, SelfRegulationSCP1*} from 4 different application domains. For comparison, we applied one classic model **1NN-DTW-D** [174] and three recently proposed semi-supervised deep learning models: **USRL** [17], **Semi-TapNet** [187] and **MTL**

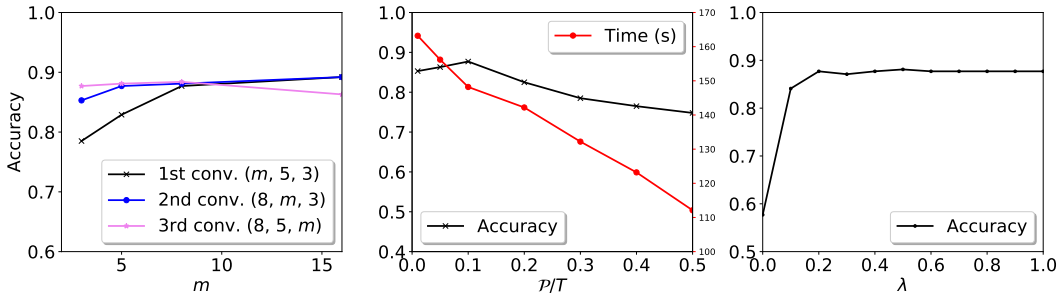


Figure 4.3: Parameter effects: *left*) the kernel/window size m at each convolutional module; *mid*) The pool size $\mathcal{P}$ for building the embedding; *right*) the hyperparameter λ which weights the regularization loss.

[178]. Since **1NN-DTW-D** and **MTL** are initially designed for UTS, we adapt them by:

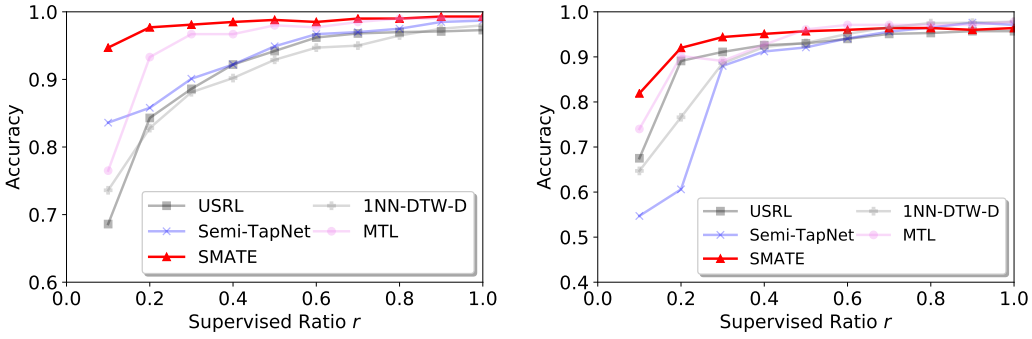
- Adopting DTW_D[4] [179] as distance in **1NN-DTW-D**.
- Updating the MTS network optimization metrics in **MTL**.

Figure 4.4 shows the classification accuracy at different supervision levels. In a Motion Recognition task (*ArticularyWorkR.*), from 10% labeled training set to fully labeled one, the accuracy of SMATE varies only by 0.046, compared to INN-DTW-D (0.264), USRL (0.286), Semi-TapNet (0.151) and MTL(0.225), showing that SMATE is capable of learning a class-separable representation under weak supervision and gives a better prediction than other classifiers under intense supervision. This conclusion is also demonstrated in EEG/MEG applications (*SelfRegulationSCP1*), with 10% labeled samples, SMATE is capable of obtaining a higher accuracy (0.781) than fully supervised 1NN-DTW-D (0.775), USRL (0.771), Semi-TapNet (0.739) and MTL (0.730). In Human Activity domain (*Epilepsy*), Semi-TapNet performs the worst with low supervised ratios, which can be explained by the fact that the limited labels restrain the intermediate-trained TapNet classifier for predicting the pseudo-labels. In an Audio Spectra task (*Heartbeat*), though the fully supervised accuracy of SMATE (0.741) is not as good as Semi-TapNet (0.751), the weakly supervised SMATE with 10% labeled samples performs the best among all semi-supervised models, indicating the reliability of the semi-supervised representation learned by SMATE.

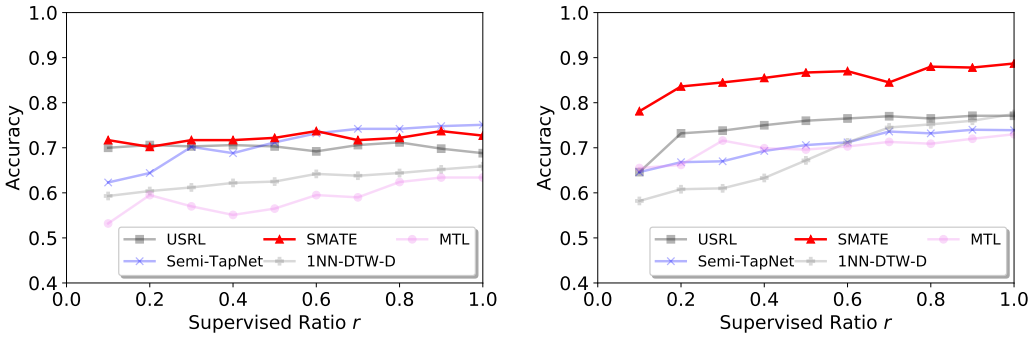
4.5.4 RQ 3: Visualization & Interpretation of the Representation Space

Apart from the thorough exploration of the weakly labeled samples, the representation space learned via SMATE shows good interpretability compared to the traditional Deep

[4] *DTW-D* and DTW_D are two different distance measures designed respectively for Univariate and Multivariate Time Series



(a) ArticularyWordR. (Motion) **(b)** Epilepsy (Human Activity)



(c) Heartbeat (Audio Spectra) **(d)** SelfRegulationSCP1 (EEG/MEG)

Figure 4.4: Semi-supervised performance comparison on datasets from different domains

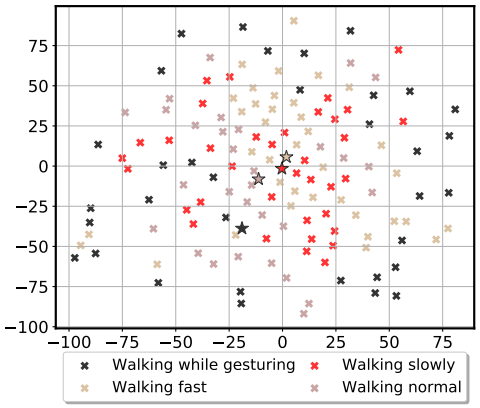
Learning models [226, 227, 125, 230] where the intermediate representations are not intuitively explainable.

We show in Figure 4.5 that t-SNE visualization of the representation space for the *Epilepsy* dataset, which contains four human activities: *Walking while gesturing*, *Walking slowly*, *Walking fast*, *Walking normally*. The 137 samples in the training set are projected to the representation space with 10% labels adopted for training. We show respectively the space visualization at each regularization step. The results suggest that: **1)** the embeddings obtained only with the autoencoder's reconstruction objective are sparsely distributed, as the auto-encoder focuses more on the overall restoration from the embeddings, but less on the inner difference between the class-specific samples; **2)** with the regularization step 1, the class centroids initialized by the 10% labeled samples tend to assemble the embeddings with the same inherent labels. The assembling ability is further enhanced by the supervised adjustment in regularization step 2. **3)** the unlabeled samples are thoroughly explored in regularization step 3 to foster the class-specific clusters, which allow building a simple but reliable classifier such as SVM. **4)** the representation space is interpretable for not only the effect of the weak supervision but also the classification results. For instance, in Figure 4.5d, three samples of *Walking while gesturing* stay close to the class centroid of *Walking fast*, which may lead to the misclassification of certain samples in the two classes. To improve the classifier, more labels of the two classes in the training set can be added.

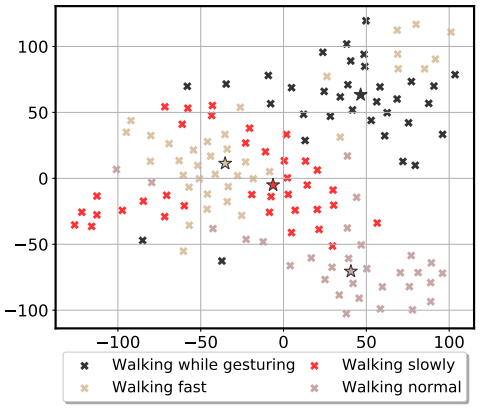
4.5.5 RQ 4: Performance of Spatial Modeling Block (SMB)

To validate the Spatial Modeling Block (SMB), firstly, we compare the classification accuracy of SMATE with or without integrating SMB on the 27 datasets that SMATE has successfully executed. Then we rebuild SMATE by replacing SMB with the following modules in the state-of-the-art work which learn the variable relationships of MTS: **Random Dimension Permutation (RDP)** in TapNet [187] and **Squeez-and-Excitation (SE)** in MLSTM-FCN [230]. Briefly, SMATE-SMB achieves [17 Wins|8 Ties|2 Losses] to SMATE-NonSMB, indicating that SMB contributes to a better MTS representation.

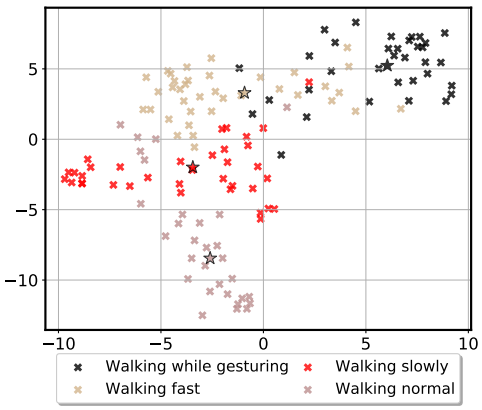
In Figure 4.6, we give a one-to-one comparison between SMB and SE/RDP on the 27 datasets. We find that SMATE performs better than other modules on modeling the spatial interactions: [14 Wins|8 Ties|5 Losses] to SE, [12 Wins|9 Ties|6 Losses] to RDP. RDP performs relatively better than SE, as a set of grouped variables produced by RDP provides various MTS views, allowing exploring the interactions between the subsets of all variables more thoroughly. However, extra parameters for variable groups are introduced. SE is a parameter-free module but considers each variable has a unique and stable state when interacting with others, which ignores the dynamic features in time series. SMB answers both the questions of the parameter-free settings and the dynamic interactions. The results show that capturing the spatial dynamic interactions at the sub-sequence level performs better than modeling the variable interactions at the sequence level [187, 230].



(a) Without any regularization



(b) Regularization step 1: supervised initialization of the class centroids



(c) Regularization step 2: supervised adjustment of the class centroids

(d) Regularization step 3: unsupervised adjustment of the class centroids

Figure 4.5: The t-SNE visualization of the representation space for the *Epilepsy* dataset. We set the supervised ratio to 0.1. The colors of the embeddings represent their inherent labels, which are not fully adopted for training. The class centroids are marked by $\star$. (a) The representation space obtained only by the reconstruction objective without any regularization. (b) Regularization step 1: 10% labeled samples in the training set are adopted to initialize the class centroids. (c) Regularization step 2: the same labeled samples are deployed to adjust the class centroids. (d) Regularization step 3: the rest unlabeled samples are deployed to adjust the class centroids.

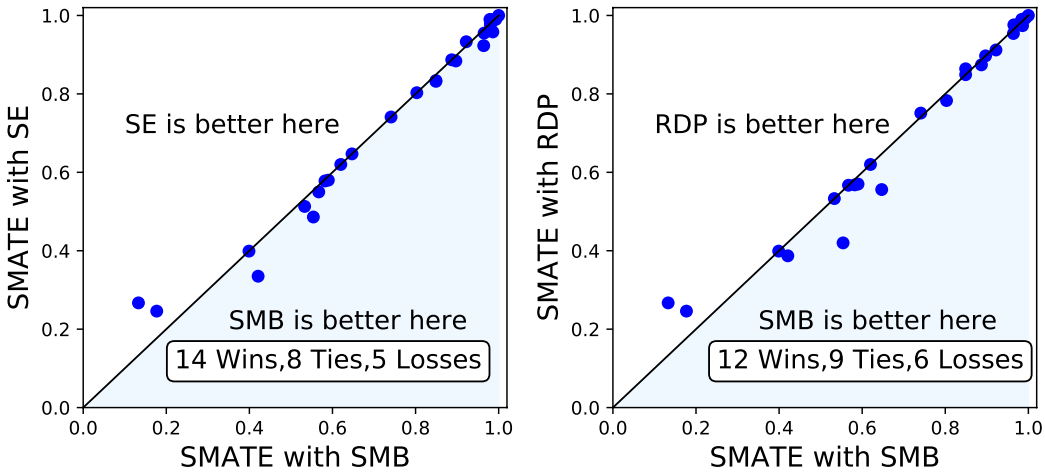


Figure 4.6: Accuracy performance comparison between *left*) SMB & SE, *right*) SMB & RDP, which are are integrated separately into SMATE.

4.5.6 RQ 5: Efficiency Analysis

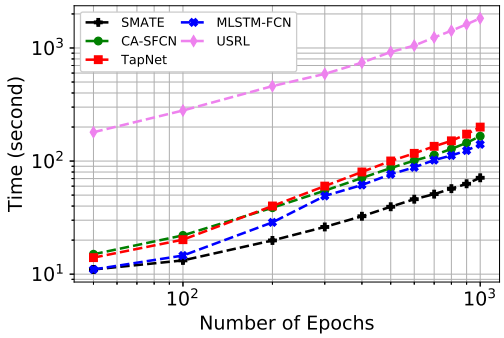
As the classical data mining classifiers do not benefit from the GPU's acceleration, it is unfair to compare them with the deep learning models on different hardware. Here, we compare SMATE with the deep learning models {MLSTM-FCN [230], CA-SFCN [125], USRL [17], and TapNet [187]}. Table 4.6 shows the models' parameter numbers on *ArticularyWordR.*. Nevertheless, they do not represent the models' efficiency.

Table 4.6: Parameter numbers of deep learning models on MTSC

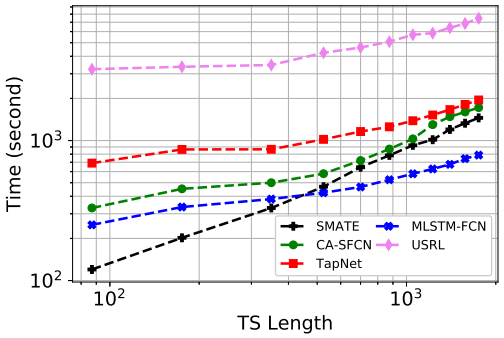
ArticularyWordR.	SMATE	MLSTM-FCN	CA-SFCN	USRL	TapNet
Param. number	564585	597865	685965	368655	635285

Figure 4.7 shows the computational efficiency concerning the different factors: (a) **the number of training epochs**, we take the dataset *ArticularyWordRecognition* with (N_{train}, M, T) = (275, 9, 144); (b) **the TS length**, we select *EthanolConcentration* with (N_{train}, M, **T**) = (261, 3, **1751**), and do random re-samplings over raw sequences; (c) **the number of TS instance** in the training set, we choose *LSST* with ($\mathbf{N_{train}}$, M, T) = (**2459**, 6, 36) and randomly re-sample the TS instances; (d) **the number of variables**, we select *PEMS-SF* with (N_{train}, **M**, T) = (267, **963**, 144) and randomly re-sample the TS variables. With the models' default parameter settings mentioned in their papers, we set 2000 training steps for USRL, 3000 training epochs for others during the tests on the factors (b)(c)(d).

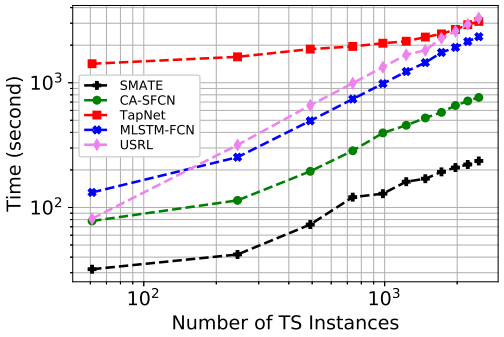
Figure 4.7 shows that the training time of the deep learning models tends to be linear in the four factors. More specifically, the results suggest that: **1)** SMATE is generally much more efficient on short TS (with $T <\sim 500$), and more costly than MLSTM-FCN



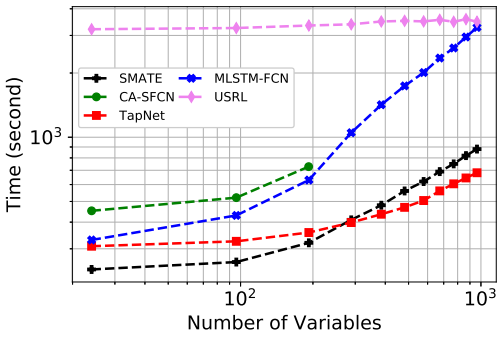
(a) *ArticularyWordRecognition*: $(N_{train},M,T) = (275,9,144)$



(b) *EthanolConcentration*: $(N_{train},M,\mathbf{T}) = (261,3,\mathbf{1751})$



(c) *LSST*: $(\mathbf{N_{train}},M,T) = (\mathbf{2459},6,36)$



(d) *PEMS-SF*: $(N_{train},\mathbf{M},T) = (267,\mathbf{963},144)$

Figure 4.7: Training time on four datasets with regard to: (a) the number of training epochs; (b) the TS length $\mathbf{T}$; (c) the number of TS instance $\mathbf{N_{train}}$; (d) the number of variables $\mathbf{M}$. We did not report the results of CA-SFCN [125] for $M > 200$ in (d) due to the memory overflow.

on long TS (see Figure 4.7b). As an MTS instance $x \in \mathcal{R}^{T \times M}$ with larger T brings a sequence embedding $h \in \mathcal{R}^{L \times D}$ with a larger L in SMATE, the regularization on a lager embedding space becomes more costly. **2)** SMATE is generally more efficient than the competitors, but tends to be more sensitive to the variable numbers **M** than USRL and TapNet (see Figure 4.7d). To explain this, first, Triplet Loss adopted in USRL requires intensive distance computations between the embeddings, which offsets the effect of larger input space. This can be also demonstrated in Figure 4.7b & 4.7c, where USRL is not sensitive to the TS length **T**, but highly sensitive to the number of TS instances $\mathbf{N_{train}}$. Second, the efficiency of SMATE is greatly affected by the input and output space of the auto-encoder, which is more sensitive to the variable numbers than the Random Dimension Permutation (RDP) block in TapNet, which works only on the input space.

Overall, USRL [17] is an order slower than SMATE due to its mini-batch optimization strategy and huge distance computations required by Triplet Loss. CA-SFCN [125] performs less than SMATE because of its costly cross attention mechanism on both temporal and spatial axes. However, MLSTM-FCN [230] and TapNet [187] outperform SMATE for long TS length ($\mathbf{T} > 500$) or huge variable numbers ($\mathbf{M} > 300$).

4.5.7 Discussion

Our approach has several advantages. First, owing to the Spatio-Temporal dynamic encoder, SMATE allows exploring more thoroughly the essential characteristics of MTS. TapNet [187] and MLSTM-FCN [230] generally consider the correlation between the entire 1-D series, while USRL [17] processes indifferently the MTS and UTS, they all ignore the fact that the interactions between 1-D segments may evolve in the dynamic sequence, which is especially important in certain domains (e.g., EEG/MEG applications).

Second, SMATE explores thoroughly the unlabeled samples, which contributes not only to the autoencoder's reconstruction objective, but also to the regularization process on the embedding space. While Semi-TapNet [187] considers unlabeled data only with the pseudo-labels predicted by intermediate-trained classifiers, which is less reliable when there are limited labels. USRL [17] trains the representation without any supervision, which shows less advantage for the classification task.

Third, the representation space learned via SMATE is interpretable for showing the effect of the three-step regularization process and explaining the classification results, which allows taking further actions to improve the classifier.

Finally, SMATE allows an efficient representation learning and classification for MTS. On the one hand, from the distance-based approaches (e.g. 1NN-based classifiers [179]) to the bag-of-patterns classifier (e.g., WEASEL+MUSE [225]), the classic data mining methods always show a high time complexity [116]. On the other hand, the deep learning-based approaches show no significant efficiency advantage to SMATE due to their larger parameter space to optimize.

4.6 Conclusion

In this Chapter, we proposed SMATE, to learn the Spatial-temporal representation on weakly-labeled multivariate time series. Different from our proposals in the last chapter, here we considered not only the temporal relationships between the observations but also the inter-variable relationships between the data sources. Specifically, inside the autoencoder-based structure, the Spatial-temporal encoder maps the temporal dynamic features and the spatial dynamic interactions into a low dimensional embedding space. A semi-supervised three-step regularization process is proposed to compel the model in learning class-separable representation. The weak supervision on the embedding space allows building a reliable classifier, which is extremely valuable in real-life scenarios with label shortage issues. The results show that the evolving variable interactions (i.e., *spatial dynamics*) play an essential role in modeling multivariate time series. Moreover, SMATE allows for visual interpretability in both the learned representation and the semi-supervised representation learning process.

A recent experimental study on MTSC models has been conducted in [116]. It concludes that MTSC is still *at an earlier stage of development than univariate TSC*. For instance, existing approaches do not consider typical features of real-world data, such as missing values and unequal length time series. Hence, our future work will be oriented towards extending SMATE to support multivariate time series with practical issues, e.g., missing values. Further, it is also possible to improve the model by, e.g., incorporating insights from CoDATS [297] with time series domain adaptation, and from ROCKET [12] with random convolutional kernels for feature extraction.

Precisely, we continued this research in the next chapter by considering both the practical Smart City scenario of MTS and the missing-value issue in the data.

Chapter 5

Geo-located Multivariate Time Series Forecasting with Missing Values

Contents

5.1 Introduction

Traffic forecasting has played a critical role in intelligent transportation systems, which helps the transportation department better manage and control traffic congestion. Generally represented by geo-located Multivariate Time Series (MTS), traffic data not only shows the typical characteristics of MTS [298], i.e., temporal dependency, but also integrates the spatial information of the traffic network, i.e., the spatial dependency between the sensor traffic nodes over the road network.

In recent years, by leveraging the spatial-temporal patterns in traffic data, many deep learning models based on recurrent neural network (RNN) [237], temporal convolutional network (TCN) [278], graph convolutional networks (GCN) [277], etc., have been applied in traffic forecasting task and achieved the state-of-the-art performance. They all have a strong assumption that the data is complete or has been well-preprocessed [130]. However, since the traffic data are generally collected from the geo-located sensors, sensor failures or communication errors will result in missing values in the collected data, which will deteriorate the performance of the forecasting model. We should remark that the missing measures are usually marked as *zero* in traffic data [277], which should be distinguished from the non-missing measures but with *zero* values. A typical example comes to the traffic flow data: no vehicles are detected during the night, then the traffic measures are marked as *zero* instead of being considered as missing.

The missing values can either be ignored in the learning model when calculating the loss function [275] or be considered before or during the training process. Apparently, ignoring the missing values hinders the model from benefiting from the rich data information for better performance, especially when the missing ratio is high. When considering the missing values in traffic data, most work [299] conduct data imputation during the preprocessing step, then import the completed data into the training step, i.e., *two-step processing*. Recent work [21, 22, 23, 24, 25] tend to jointly consider the missing values and the forecasting modeling during the training step (i.e., *one-step processing*) and declared a better performance than the two-step processing. However, the above-mentioned work suffer from three major issues. First, the missing and zero values are usually considered indifferent, leading to unnecessary, even harmful data imputations, thus contradicting the raw data information. Second, most of the work [21, 22, 24, 25] considers missing values from the temporal aspect, ignoring the rich information from the spatial perspective. Third, they are generally designed for processing the missing values in some basic scenarios, such as random missing or temporal block missing, but lack of power for the complex scenarios as shown in Fig. 5.1. In the real-world, the missing values in traffic data occur on both long-range (e.g., *device power-off*) and short-range (e.g. *device errors*) settings, on partial (e.g., *local sensor errors*) and entire transportation network (e.g., *control center errors*). The complex scenarios require a holistic approach for handling various types of missing values together.

Therefore, to handle both the Spatio-temporal patterns and complex missing-value

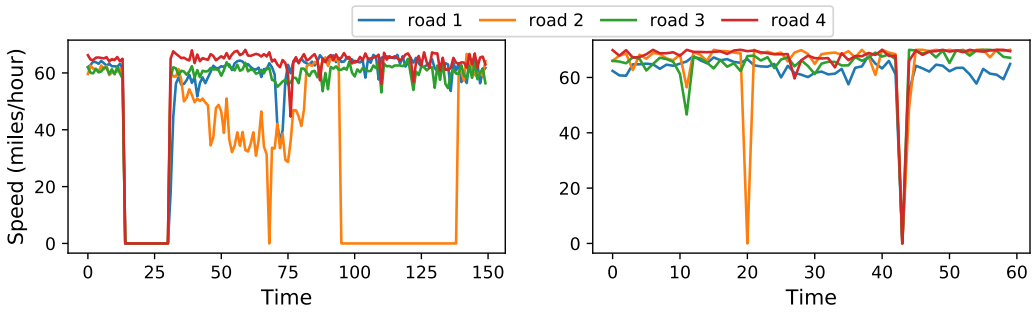


Figure 5.1: Missing measures of traffic speed data from METR-LA dataset [237]. *left*) long-range missing on entire network (i.e., Spatio-temporal block) and partial network (i.e., temporal block); *right*) short-range random missing on partial network (i.e., temporal values) and entire network (i.e., Spatio-temporal vectors).

scenarios in traffic data, we propose GCN-M, **G**raph **C**onvolutional **N**etworks for Traffic Forecasting with **M**issing Values. The graph neural network-based structure allows jointly modeling the Spatio-temporal patterns and the missing values in one-step processing. We construct local statistic features from spatial and temporal perspectives for handling short-range missing values, which is further enhanced by a memory module to extract global historical features for processing long-range missing blocks. The combined local-global features allow not only identifying the missing measures from the inherent zero values but also enriching the traffic embeddings, thus generating dynamic traffic graphs to model the dynamic spatial interactions between traffic nodes. The missing values on a partial and entire network can then be considered from spatial and temporal perspectives.

We summarize our main contributions as follows:

- **Complex missing value modeling**: We study the complex scenario where missing traffic values occur on both short & long ranges and on partial & entire transportation network.

- **Spatio-temporal memory module**: We propose a memory module that can be used by GCN-M to learn both local Spatio-temporal features and global historical patterns in traffic data for handling the complex missing values.

- **Dynamic graph modeling**: We propose a dynamic graph convolution module that models the dynamic spatial interactions. The dynamic graph is characterized by the learned local-global features at each timestamp, which not only offset the missing values' impact but also help learn the graph.

- **Joint model optimization**: We jointly model the Spatio-temporal patterns and missing values in one-step processing, which allows processing missing values specifically for traffic forecasting tasks, thus bringing better model performance than two-step processing.

- **Extensive experiments on real-life data**: The experiments are carried out on two real-life traffic datasets. We provide detailed evaluations with 12 baselines, which show the effectiveness of GCN-M over the state-of-the-art.

The rest of this chapter starts with a review of the most related work. Then, we formulate the research problems of the chapter. Later, we present in detail our proposal GCN-M, followed by the experiments on real-life datasets and the conclusion.

5.2 Related Works

5.2.1 Graph Convolutional Networks for Traffic Forecasting

Graph Convolutional Network (GCN) is a special kind of Convolutional Neural Network (CNN) generalized for graph-structured data. Most of the GCN-related work focuses on graph representation, which learns node embedding by integrating the features from the node's local neighbors based on the given graph structure, i.e., adjacency matrix. The traffic data shows strong dependencies between the spatial nodes, for which GCN can be naturally suitable. Various work [237, 276, 130, 275] empowered by GCN achieved remarkable performance when doing traffic forecasting tasks. However, they either rely on spatial and temporal completion of the data or calculate loss function for non-zero entries, i.e., only calculate the loss on entries that contain valid sensor readings, which may introduce derivations when modeling the Spatio-temporal relations between the sensor nodes. In other words, missing values may hinder the traffic graph learning [277], especially for dynamic graph learning [281] where non-missing measures are required to characterize the dynamic graph at each timestamp.

5.2.2 Missing value processing

The simplest solution of processing missing values in MTS would be data imputation such as statistic imputation (e.g., mean, median), EM-based imputation, K-nearest neighborhood, and matrix factorization. It's generally believed that those methods fail to model temporal dynamics of time series [25]. In other words, they are not applicable for handling long-range missing values. Recent generative models [300] show reliable performance for long-range time series imputation. However, isolating the imputation model from the forecasting model leads to a two-step processing and may generate sub-optimal results [299]. To handle this issue, recent studies [21, 25] jointly model the missing values and forecasting task in one-step processing. For instance, GRU-D [21] considers the nearby temporal statistical features to do imputations inside GRUs, whereas LSTM-I [22] infers missing values at current time step from preceding LSTM cell states and hidden states, SGMN [23] improved the state transition process via a Graph Markov Process. Limited to short-period missing context, those methods are further enhanced by LGnet [25] with the

global temporal dynamics to handle the long-range missing issue, and by LSTM-M [24] with multi-scale modeling to better explore historical information. However, the above-mentioned models handle missing values with a focus on the temporal aspect without considering the complex Spatio-temporal features in traffic data. Specifically, the strong spatial connections between the sensor nodes should provide us with more information to handle the missing values. Moreover, the one-step processing models are generally designed for single-step forecasting without considering the multi-step settings. Table 5.1 shows the method comparison for traffic forecasting with missing values.

Table 5.1: Existing methods for Traffic Forecasting with missing values

	Imputation-based	GRU-D	LSTM-I	LSTM-M	LGnet	SGMN	**GCN-M**
Short-range missing	✓	✓	✓	✓	✓	✓	✓
Long-range missing	-	-	-	✓	✓	-	✓
Multi-scale modeling	-	-	-	✓	-	-	✓
Spatial modeling	-	-	-	-	-	✓	✓
One-step processing	-	✓	✓	✓	✓	✓	✓
Multi-step forecasting	✓	-	-	-	✓	-	✓

5.3 Problem Formulation

We aim to predict the traffic data in the future by leveraging historical traffic data. The traffic data can be represented as multivariate time series on the traffic network. Let the traffic network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, ..., v_N\}$ is a set of N traffic sensor nodes and $\mathcal{E} = \{e_1, ..., e_E\}$ is a set of E edges connecting the nodes. Each node contains F features representing traffic flow, speed, occupancy, etc. We use $\mathcal{X}=\{\mathbf{X}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times F\times\tau}$ to denote all the feature values of all the nodes over τ time slices, $\mathbf{X}_t = (\mathbf{x}_t^1, ..., \mathbf{x}_t^N) \in \mathcal{R}^{N\times F}$ denotes the observations at time t, where $\mathbf{x}_t^i \in \mathcal{R}^F$ is the i-th variable of X_t. We define a mask sequence $\mathcal{M}=\{\mathbf{M}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times F\times\tau}$, $\mathbf{M}_t = (\mathbf{m}_t^1, ..., \mathbf{m}_t^N) \in \mathcal{R}^{N\times F}$, $\mathbf{m}_t^i \in \{0, 1\}^F$ denotes the features' missing status for the i-th variable. To simplify, we adopt $x_t^i \in \mathcal{R}$ and $m_t^i \in \mathcal{R}$ to denote respectively the observation and mask value of *one single feature* for the i-th variable of $\mathbf{X}_t$. We take $m_t^i = 0$ if x_t^i is missing, otherwise $m_t^i = 1$.

We aim to build a model f, which can take an incomplete traffic sequence $\{\mathcal{X}, \mathcal{M}\}$ and the traffic network $\mathcal{G}$ as input, to predict the traffic data for the next T_p time steps $\mathbf{Y} = \{y_{\tau+1}, ..., y_{\tau+T_p}\} \in \mathcal{R}^{N\times T_p}$.

5.4 Proposal: GCN-M

The traffic data are collected under complex urban conditions. Apart from the Spatio-temporal patterns in the traffic data, we also consider the scenarios of complex missing values. We design a solution that models the local Spatio-temporal features and global

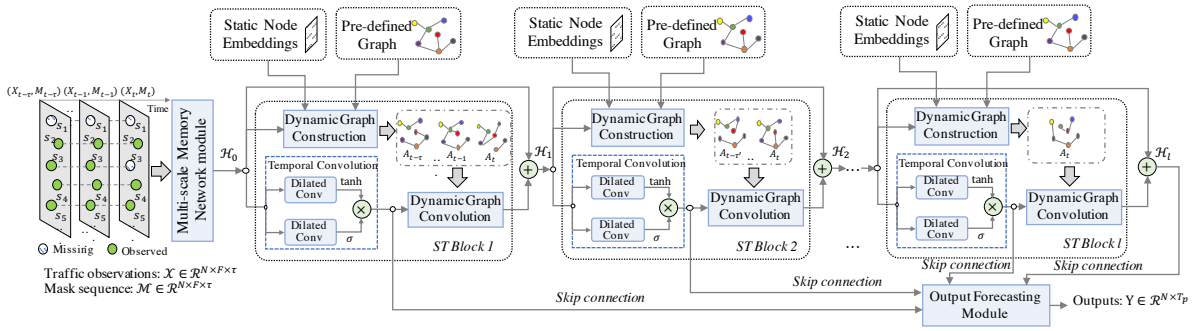


Figure 5.2: Main architecture of GCN-M

historical patterns in a dynamic manner. The complex missing values are considered when building the forecasting model, i.e., one-step processing.

5.4.1 Model Architecture

The global structure of GCN-M is shown in Fig. 5.2, integrating a Multi-scale Memory Network module, an Output Forecasting module, and l Spatio-Temporal (ST) blocks. Each ST block integrates three key components: Temporal Convolution, Dynamic Graph Construction, and Dynamic Graph Convolution. The input traffic observations $\mathcal{X} \in \mathcal{R}^{N\times F\times\tau}$ and the mask sequence $\mathcal{M} \in \mathcal{R}^{N\times F\times\tau}$ are fed into the multi-scale memory network to extract the local statistic features and global historical patterns thus enriching the traffic embeddings. On the one hand, the enriched embeddings $\mathcal{H}_i$ on each ST block are used to mark the dynamic traffic status, thus generating dynamic graphs by combining both static node embeddings and predefined graph information. On the other hand, the learned dynamic graphs are combined with the temporal convolution module via a dynamic graph convolution to capture temporal and spatial dependencies in the traffic embeddings. We adopt residual connections between the input and output of each ST block to avoid the gradient vanishing problem. The output forecasting module takes the skip connections on the output of the final ST block and the hidden states after each temporal convolution for final prediction.

5.4.2 Multi-scale Memory Network

To extract the local statistic features and global historical patterns then form an enriched embedding, we adopt the concept of memory network, which was firstly proposed in [301] with primary application in Question-Answer (QA) systems. As shown in Fig. 5.3, the main idea of our memory network is to learn from historical memory components which conserve the long-range multi-scale patterns, i.e., recent, daily-periodic, and weekly-periodic dependencies. The scale range depends on the data characteristics. Specifically, we first extract local Spatio-temporal features as keys to query the memory components;

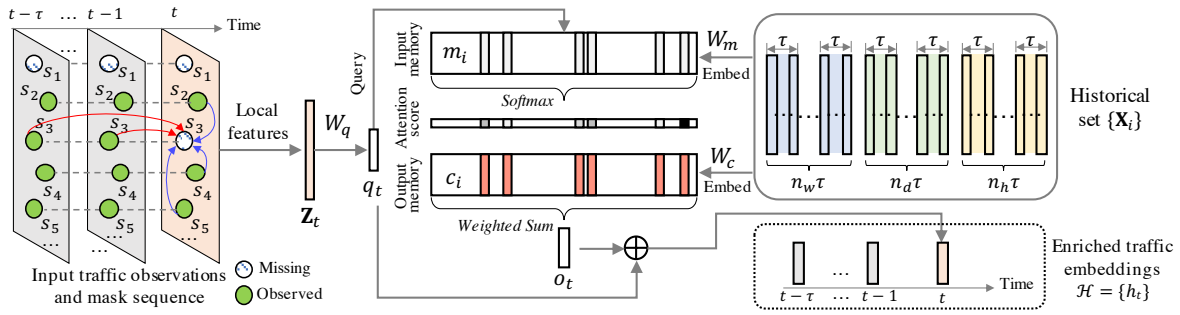


Figure 5.3: Memory module enriches traffic embeddings with multi-scale global features

the weighted historical long-range patterns will be cooperated with the local statistic features to eliminate the side effect from the missing values. Then, the local-global features will be output as the enriched traffic embeddings.

5.4.2.1 Local Spatio-temporal features

We first extract the Spatio-temporal features using the contextual information from observed parts of the time series. Unlike prior studies [21], we consider both temporal and spatial aspects for generating the following statistic features of every timestamp:

Empirical Temporal Mean: The mean of previous observations reflects the recent traffic state and serves as a contextual knowledge of x_t^i. Therefore, for a missing value $x_t^i \in \mathcal{R}$, we construct its temporal mean using L observed samples x_*^i before time t:

$$\bar{x}_t^i = \sum_{l=t-L}^{t-1} m_l^i x_l^i / \sum_{l=t-L}^{t-1} m_l^i \tag{5.1}$$

Last Temporal Observation: we adopt the assumption in [21] that any missing value inherits more or less the information from the last non-missing observation, in other words, the temporal neighbor stays close to the current missing value. We use $\dot{x}_t^i$ to denote the last temporal observation of x_t^i, their temporal distance is defined as $\dot{\delta}_t^i$.

Empirical Spatial Mean: Another contextual knowledge of x_t^i is from the nearby nodes, which reflects the current local traffic situation. For each missing value x_t^i, we construct its empirical spatial mean using S observed samples x_t^* nearby the sensor node i:

$$\bar{\bar{x}}_t^i = \sum_{s=1}^{S} m_t^s x_t^s / \sum_{s=1}^{S} m_i^s \tag{5.2}$$

Nearest Spatial Observation: typically, the state of a graph node remains relatively similar to its neighbors, especially in a traffic graph where the nearby nodes share similar

traffic situation. We define $\ddot{x}_t^i$ as the nearest spatial observation of x_t^i, their spatial distance is denoted as $\ddot{\delta}_t^i$.

Generally, when $\dot{\delta}_t^i$ or $\ddot{\delta}_t^i$ is smaller, we tend to trust $\dot{x}_t^i$ or $\ddot{x}_t^i$ more. When the spatial/temporal distance becomes larger, the spatial/temporal mean would be more representative. Under this assumption, we model the temporal and spatial decay rate γ as

$$\gamma_t(\dot{\delta}_t^i) = exp\{-max(0, w^i\dot{\delta}_t^i + b^i\} \tag{5.3}$$

$$\gamma_s(\ddot{\delta}_t^i) = exp\{-max(0, w_t\ddot{\delta}_t^i + b_t\} \tag{5.4}$$

where w^i, w_t, b^i and b_t are model parameters that we train jointly with other parameters of the traffic forecasting model. We chose the exponentiated negative rectifier [21] so that the decay rates γ_t and γ_s decrease monotonically in the range between 0 and 1. Considering the trainable decays, our proposed model incorporates the spatial/temporal estimations to define the local features of x_t^i:

$$z_t^i = m_t^i x_t^i + (1 - m_t^i)(\gamma_t\dot{x}_t^i + \gamma_s\ddot{x}_t^i + (1 - \gamma_t)\bar{x}_t^i + (1 - \gamma_s)\bar{\bar{x}}_t^i) \tag{5.5}$$

Therefore, for $\mathbf{X}_t \in \mathcal{R}^{N\times F}$, we can get its local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$.

5.4.2.2 Multi-scale Memory Construction

The global historical patterns play a critical role in building an enriched traffic embedding. The historical observations in multiple scales (e.g., hourly, daily, weekly) can be embedded into memory as complement information for the local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$. The main idea is to adopt local features to query similar historical patterns in the memory and output a weighted feature representation for the current timestamp. In this manner, the enriched multi-scale historical and local features allow not only eliminating the side effect of missing values but also improving the current feature embeddings. At time t, the query q_t of $\mathbf{X}_t$ can be embedded from the local features $\mathbf{Z}_t \in \mathcal{R}^{N\times F}$:

$$q_t = \mathbf{Z}_t W_q + b_q \in \mathcal{R}^{N\times d} \tag{5.6}$$

where $W_q \in \mathcal{R}^{F\times d}$, $b_q \in \mathcal{R}^{N\times d}$ are parameters, d is the embedding dimension.

The input memory components are the temporal segments of multiple scales:

- The recent (e.g., hourly) segment is: $X_h = \{\mathbf{X}_i\}_{i=t-\tau}^{t-1} \in \mathcal{R}^{N\times F\times n_h\tau}$, with n_h recent periods (e.g., hours) before t, each period contains τ observations.
- The daily-periodic segment is: $X_d = \{\mathbf{X}_i\} \in \mathcal{R}^{N\times F\times n_d\tau}$ with $i \in [t-n_dT_d-\tau/2 : t-n_dT_d+\tau/2] \parallel [t-(n_d-1)T_d-\tau/2 : t-(n_d-1)T_d+\tau/2] \parallel ... \parallel [t-T_d-\tau/2 : t-T_d+\tau/2]$, we store τ samples around time t for each of the past n_d days. T_d denotes the sample number during one day, $\parallel$ indicates the concatenation operation.

- The weekly-periodic segment is: $X_w = \{\mathbf{X}_i\} \in \mathcal{R}^{N\times F\times n_w\tau}$ with $i \in [t - n_wT_w - \tau/2 : t - n_wT_w + \tau/2] \parallel [t - (n_w - 1)T_w - \tau/2 : t - (n_w - 1)T_w + \tau/2] \parallel ... \parallel [t - T_w - \tau/2 : t - T_w + \tau/2]$, we store τ samples around time t for each of the past n_w weeks. T_w denotes the sample number during one week, $\parallel$ indicates the concatenation operation.

The input set of $\{\mathbf{X}_i\} = [X_h \| X_d \| X_w] \in \mathcal{R}^{N\times F\times(n_d+n_w+n_h)\tau}$ are embedded into the input memory vectors $\{m_i\}$ and output memory vectors $\{c_i\}$:

$$m_i = \mathbf{X}_i W_m + b_m \in \mathcal{R}^{N\times d} \tag{5.7}$$

$$c_i = \mathbf{X}_i W_c + b_c \in \mathcal{R}^{N\times d} \tag{5.8}$$

where $W_m, W_c \in \mathcal{R}^{F\times d}$, $b_m, b_c \in \mathcal{R}^{N\times d}$ are parameters.

In the embedding space, we compute the attention score between the query q_t and each memory m_i by taking the inner product followed by a softmax:

$$p_{t,i} = Softmax(q_t^T m_i) \tag{5.9}$$

The attention score represents the similarity of each historical observation to the query. Any pattern with a higher attention score is more similar to the context of targeting missing value. As shown in Fig. 5.3, the response vector from memory is then a sum over the output memory vectors, weighted by the attention score from the input:

$$o_t = \sum_{i=1}^{(n_d+n_w+n_h)\tau} c_i p_{t,i} \in \mathcal{R}^{N\times d} \tag{5.10}$$

We can finally integrate both local Spatio-temporal and global multi-scale features and output the enriched traffic embeddings:

$$h_t = (q_t \| o_t) W_h + b_h \in \mathcal{R}^{N\times d} \tag{5.11}$$

where $W_h \in \mathcal{R}^{2d\times d}$, $b_h \in \mathcal{R}^d$ are parameters, and $\|$ denotes the concatenation operation. Therefore, for input $\mathcal{X} = \{\mathbf{X}_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times F\times\tau}$, we can get its enriched traffic embeddings $\mathcal{H} = \{h_t\}_{t=1}^{\tau} \in \mathcal{R}^{N\times d\times\tau}$.

5.4.3 Dynamic Graph Construction

A predefined graph is usually constructed with the distance or the connectivity between the spatial nodes. However, recent studies [275, 276, 277] show that the cross-region dependence does exist for those nodes which are not physically connected but share similar patterns. Learning dynamic graphs should probably show better performance than learning static graphs or adopting the predefined graphs. Considering the missing values in traffic data, instead of using the raw traffic observations to mark the dynamic traffic status [277, 302], we construct dynamic graphs (i.e., adjacency matrix) with the enriched traffic

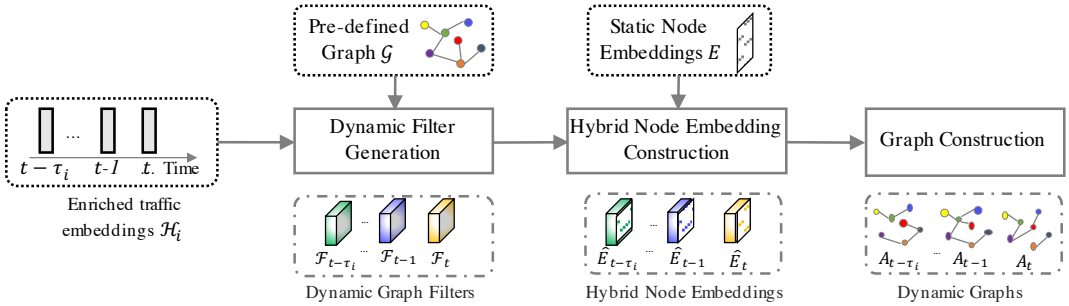


Figure 5.4: Dynamic Graph Construction from the enriched traffic embeddings

embeddings $\mathcal{H}_i$ at each ST block, which integrates both local and global multi-scale patterns at each time step, allowing capturing the spatial relationship between traffic nodes robustly. As shown in Fig. 5.4, the main idea here is to generate dynamic filters from the predefined graphs $\mathcal{G}$ and the traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N\times d\times \tau_i}$ (τ_i is the sequence length at the i-th ST block), which are applied on the static randomly initialized node embeddings to construct dynamic adjacency matrix. In more detail, the core steps in Fig. 5.4 are illustrated as follows:

[Dynamic Filter Generation] Given $\mathcal{H}_i = \{h_t\} \in \mathcal{R}^{N\times d\times \tau_i}$, the traffic embedding h_t at time t is firstly combined with the predefined adjacency matrix $A_{\mathcal{G}} \in \mathcal{R}^{N\times N}$ to generate dynamic graph filters via a diffusion convolution layer as proposed in [237]:

$$\mathcal{F}_t = \sum_{k=0}^{K} P_k h_t W_k \in \mathcal{R}^{N\times d} \tag{5.12}$$

where K denotes the diffusion step, $P_k = A_{\mathcal{G}}/rowsum(A_{\mathcal{G}})$ represents the power series of the transition matrix [278], and $W_k \in \mathcal{R}^{d\times d}$ is the model parameter matrix.

[Hybrid Node Embedding Construction] Considering both the source and target traffic node, we initialise two random node embeddings $E^1, E^2 \in \mathcal{R}^{N\times d}$, representing the static node features [275] which are not reflected in the observations but learnable during training. Thus, two dynamic filters are applied over the static node embeddings:

$$\begin{aligned} \hat{E}_t^1 &= tanh(\alpha(\mathcal{F}_t^1 \odot E^1)) \in \mathcal{R}^{N\times d} \\ \hat{E}_t^2 &= tanh(\alpha(\mathcal{F}_t^1 \odot E^2)) \in \mathcal{R}^{N\times d} \end{aligned} \tag{5.13}$$

where $\odot$ denotes the Hadamard product [278]. $\hat{E}_t^1$ and $\hat{E}_t^1$ are hybrid node embeddings combining both static and dynamic settings of the traffic data.

[Graph Construction] Following [276], we extract uni-directional relationships between traffic nodes. The dynamic adjacency matrix is constructed from the hybrid embeddings:

$$A_t = ReLU(tanh(\alpha(\hat{E}_t^1 \hat{E}_t^{2^T} - \hat{E}_t^2 \hat{E}_t^{1^T}))) \in \mathcal{R}^{N\times N} \tag{5.14}$$

Therefore, we can construct the dynamic graphs $A_{\mathcal{D}_i} = \{A_t\} \in \mathcal{R}^{N\times N\times \tau_i}$ for the enriched traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N\times d\times \tau_i}$ at the i-th ST block.

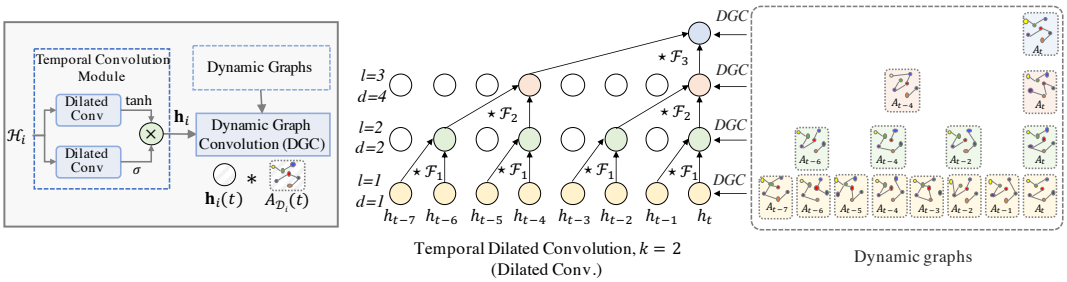


Figure 5.5: Temporal Convolution module with Dynamic Graph Convolution

5.4.4 Temporal Convolution Module

The temporal convolution network (TCN) [129] consists of multiple dilated convolution layers, which allows extracting high-level temporal trends. Compared to RNN-based approaches, dilated causal convolution networks are capable of handling long-range sequences in a parallel manner. The output of the last layer is a representation that captures temporal dynamics in history. As shown in Fig. 5.5, considering the temporal dynamics in traffic data, we adopt the temporal convolution module [278] with the consideration of the gating mechanism over the enriched traffic embeddings $\mathcal{H}_i$. One dilated convolution block is followed by a tangent hyperbolic activation function to output the temporal features. The other block is followed by a sigmoid activation function as a gate to determine the ratio of information that can pass to the next module.

Given the enriched traffic embeddings $\mathcal{H}_i = \{h_t\} \in \mathcal{R}^{N \times d \times \tau_i}$, a filter $\mathcal{F} \in \mathcal{R}^{1 \times \mathrm{K}}$, K is the temporal filter size, $\mathrm{K} = 2$ by default. The dilated causal convolution operation of $\mathcal{H}_i$ with $\mathcal{F}$ at time t is represented as:

$$\mathcal{H}_i \star \mathcal{F}_i(t) = \sum_{s=0}^{\mathrm{K}} \mathcal{F}_i(s)\mathcal{H}_i(t - \mathbf{d} \times s) \in \mathcal{R}^{N \times d \times \tau_{i+1}} \tag{5.15}$$

where $\star$ is the convolution operator, $\mathbf{d}$ is the dilation factor, d is the embedding dimension size, τ_{i+1} is the new sequence length after the convolution operation, which equals to one on the last layer. Fig. 5.5 shows a three-layer dilated convolution block with $\mathrm{K} = 2$, $\mathbf{d} \in [1, 2, 4]$. Considering the gating mechanism, we define the output of the temporal convolution module:

$$\mathbf{h}_i = tanh(W_{\mathcal{F}^1} \star \mathcal{H}_i) \odot \sigma(W_{\mathcal{F}^2} \star \mathcal{H}_i) \in \mathcal{R}^{N \times d \times \tau_{i+1}} \tag{5.16}$$

where $W_{\mathcal{F}^1}$, $W_{\mathcal{F}^2}$ are learnable parameters of convolution filters, $\odot$ denotes the element-wise multiplication operator, $\sigma(\cdot)$ is the sigmoid function.

A classic temporal convolution module stacks the temporal features at each time step t. Therefore, the upper layer contains richer information than the lower layer. The gating mechanism allows filtering the temporal features on the lower layers by weighting features on different time steps without considering the spatial node interactions at each time step. Moreover, the spatial interactions in traffic data always show a dynamic nature [276].

To this end, the gating mechanism from a dynamic spatial aspect is envisaged to better capture the Spatio-temporal patterns.

5.4.5 Dynamic Graph Convolution

Spatial interactions between the traffic nodes could be used to improve traffic forecasting performance. The dynamic spatial interaction leads to considering a dynamic version of graph convolution to conduct it on different graphs at different timestamps. Different from previous work [277] which uses raw traffic observations to mark the dynamic traffic status, we adopt the enriched traffic embeddings, which consider the missing-value issues to generate robust dynamic graphs.

As shown in Fig. 5.5, we apply the dynamic graph convolution on $\mathbf{h}_i$, i.e., the output of the temporal convolution module, to further select the features at each time step from the spatial perspective. As mentioned in Section 5.4.3, the dynamic graphs $A_{\mathcal{D}_i} \in \mathcal{R}^{N\times N\times \tau_i}$ are generated from the enriched traffic embeddings $\mathcal{H}_i \in \mathcal{R}^{N\times d\times \tau_i}$ at the i-th ST block. A_t reflects the spatial relationships between nodes at time t. The temporal features $\mathbf{h}_i(t)$ aggregate spatial information according to the adjacency matrix A_t. Inspired by DCRNN [237], we consider the traffic situation as the diffusion procedure on graph. The graph convolution will generate the aggregated spatial information at each time step:

$$\mathcal{H}'_i(t) = \sum_{k=0}^{K} \left(A_{\mathcal{D}_i}(t)\right)^k \mathbf{h}_i(t) W_k \in \mathcal{R}^{N\times d} \tag{5.17}$$

where K denotes the diffusion step, W_k is the learnable parameter matrix. We adopt the residual connection [118] between the input and output of each ST block to avoid the gradient vanishing issue in model's training. Therefore, the input of the $(i+1)^{th}$ ST block is defined as:

$$\mathcal{H}_{i+1}(t) = \mathcal{H}_i(t) + \mathcal{H}'_i(t) \tag{5.18}$$

5.4.6 Output Forecasting Module

The outputs $\mathbf{h}_i \in \mathcal{R}^{N\times d\times \tau_{i+1}}$ of the middle temporal convolution modules and $\mathcal{H}_l \in \mathcal{R}^{N\times d\times 1}$ of the last ST block are considered for the final prediction, which represent the hidden states at various Spatio-temporal levels. We add skip connection on each of the hidden states which are essentially $1 \times \tau_{i+1}$ standard convolutions, τ_{i+1} denotes the sequence length at the output of the i-th ST block. the concatenated output features are defined as follows:

$$O = (\mathbf{h}_0 W_s^0 + b_s^0) \| ... \| (\mathbf{h}_i W_s^i + b_s^i) \| ... \| (\mathbf{h}_{l-1} W_s^{l-1} + b_s^{l-1}) \ \| (\mathcal{H}_l W_s^l + b_s^l) \tag{5.19}$$

where $O \in \mathcal{R}^{N\times ld}$, W_s^i, b_s^i are learnable parameters for the convolutions. Two fully-connected layers are added to project the concatenated features into the desired output dimension:

$$\hat{\mathbf{Y}} = W_{fc}^2 (W_{fc}^1 O + b_{fc}^1) + b_{fc}^2 \in \mathcal{R}^{N\times T_p} \tag{5.20}$$

where W_{fc}^1, W_{fc}^2, b_{fc}^1, b_{fc}^2 are learnable parameters for the fully-connected layers, N is the node number, T_p denotes the forecasting steps.

Given the ground truth $\mathbf{Y} \in \mathcal{R}^{N \times T_p}$ and the predictions $\hat{\mathbf{Y}} \in \mathcal{R}^{N \times T_p}$, we use mean absolute error (MAE) as our model's loss function for training:

$$L = \frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n| \tag{5.21}$$

5.5 Experiments

In this section, we demonstrate the effectiveness of GCN-M [1] with real-life traffic datasets. The experiments were designed to answer the following research questions (RQs):

RQ 1 *Performance on complete datasets:* How well our model performs on traffic datasets without missing values?

RQ 2 *Complex scenarios of missing values:* How successful is our model in forecasting traffic data considering the complex missing values scenarios?

RQ 3 *Dynamic graph modeling:* How our method performs on dynamic graph modeling considering the missing values?

5.5.1 Experimental settings

[Datasets] We base our experiments on the public traffic datasets: PEMS-BAY and METR-LA released by [237], which are widely used in the literature. PEMS-BAY records six months of traffic speed on 325 sensors in the Bay Area. METR-LA records four months of traffic flow on 207 sensors on the highways of Los Angeles County. Both datasets contain some zero/missing values, though PEMS-BAY has been pre-processed by the domain experts from the data provider [303] to interpolate most of the missing values. Following [237], the datasets are split with 70% for training, 10% for validation and 20% for testing. We use recent τ= 12 timestamps as input to predict the next T_p timestamp. Considering that the missing values are marked as zeros, we scale the input by dividing with the max speed of the training set instead of applying Z-score normalization, which avoids changing the zero values and removing the missing markers. Table 5.2 shows the summary statistics of the datasets.

[Evaluation metrics] The forecasting accuracy of all tested models is evaluated by three metrics, including mean absolute error (MAE), root mean square error (RMSE) and mean

[1] The source code is publicly available in https://github.com/GCN-M/GCN-M

Table 5.2: Summary statistics of PEMS-BAY and METR-LA

Data	#Nodes	#Edges	Length	Sample Rate	Observations	Zero ratio
PEMS-BAY	325	2369	52 116	5 mins	16 937 179	0.0031%
METR-LA	207	1515	34 272	5 mins	6 519 002	8.11%

absolute percentage error (MAPE).

$$\begin{aligned} MAE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n| \\ RMSE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \sqrt{\frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n|^2} \\ MAPE(\mathbf{Y}, \hat{\mathbf{Y}}) &= \frac{1}{NT_p} \sum_{n=1}^{N} \sum_{t=1}^{T_p} |\frac{\hat{\mathbf{Y}}_t^n - \mathbf{Y}_t^n}{\mathbf{Y}_t^n}| \end{aligned} \tag{5.22}$$

where N denotes the node numbers, T_p represents the forecasting steps.

[Execution and Parameter Settings] The proposed model is implemented by PyTorch 1.6.0 and is trained using the Adam optimizer with a learning rate of 0.001. All the models are tested on a single Tesla V100 GPU of 32 Go memory. In the multi-scale memory module, L, S are set to 12 and 5. n_h, n_d, n_w are all set to 2. We apply four ST blocks in which the Temporal Convolution module contains two dilated layers with dilation factor $\mathbf{d} \in [1, 2]$. The embedding dimension d is set to 32.

5.5.2 Baseline Approaches

We only compare with the baseline models whose source code is publicly available. We follow the default parameter settings described in each paper for training. According to the strategy for handling missing values, the baseline models can be organized into two categories:

1. Jointly model the missing values and forecasting task, i.e., one-step processing
    - GRU [304]: Gated Recurrent Unit (GRU) can be considered as a basic structure for traffic forecasting.
    - GRU-I [21]: A variation of GRU, which infers the missing values with the predictions from previous steps.
    - GRU-D [21]: Based on GRU, GRU-D incorporates the missing patterns, including the masking information and time intervals between missing and observed values, to help improve the prediction performance.
    - LSTM-I [23]: Based on LSTM, LSTM-I is similar to GRU-I for inferring the missing values.

- LSTM-M [24]: Based on LSTM, LSTM-M is designed for traffic forecasting on data with short-period and long-period missing values.
- SGMN [23]: Based on the graph Markov process, SGMN incorporates a spectral graph convolution to do traffic forecasting on data with random missing values.

2. Ignore the missing values when optimizing the model:

   - DCRNN [237]: Based on the predefined graphs, DCRNN integrates GRU with dual directional diffusion convolution.
   - STGCN [130]: Based on the predefined graphs, STGCN combines graph convolution into 1D convolutions.
   - Graph WaveNet [278]: Graph WaveNet learns an adaptive graph and integrates diffusion graph convolutions with temporal convolutions.
   - MTGNN [276]: MTGNN learns an adaptive graph and integrates mix-hop propagation layers in the graph convolution module. Moreover, it designed the dilated inception layers in temporal convolutions.
   - AGCRN [279]: AGCRN learns an adaptive graph and integrates with recurrent graph convolutions with node adaptive parameter learning.
   - GTS [280]: GTS learns a probabilistic graph which is combined with the recurrent graph convolutions to do traffic forecasting.

     Note: In practice, any model can ignore the missing values in their optimization process. We list here some classic models and the most recent models designed specifically for traffic forecasting.

5.5.3 RQ 1: Performance on complete datasets

Recently, a lot of traffic forecasting models [250] have been proposed, achieving remarkable performance on the benchmark datasets PEMS-BAY and METR-LA. Our objective is not to beat all the models in terms of forecasting accuracy but to validate our proposal for jointly modeling missing values and forecasting. Therefore, it's essential to know how GCN-M performs in a primary setting, i.e., on the original datasets without or with a few missing values. We pick three classic models (DCRNN [237], STGCN [130] and Graph WaveNet [278]) and three most recent models (MTGNN [276], AGCRN [279] and GTS [280]), which focus on the Spatio-temporal modeling of traffic data and generally ignore the missing values when training the model. We consider as well the group of works [21, 23, 24] which are specifically designed for modeling the missing values in the forecasting model, i.e., one-step processing models.

Table 5.3 and 5.4 show the performance comparison on the complete PEMS-BAY and METR-LA datasets, respectively. It should be noted that the original datasets already contain missing values (0.0031% missed in PEMS-BAY, 8.11% missed in METR-LA). We

Table 5.3: Performance comparison on **complete** PEMS-BAY dataset

PEMS-BAY	Horizon=1 (5 mins)			Horizon=3 (15 mins)			Horizon=6 (30 mins)			Horizon=12 (60 mins)		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	0.96	1.63	1.81%	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	0.98	1.84	1.98%	1.44	2.88	3.16%	1.85	3.82	4.20%	2.21	4.52	5.09%
GraphWaveNet	0.91	**1.56**	1.72%	**1.31**	2.75	**2.73%**	1.65	3.75	3.74%	1.99	4.62	4.78%
MTGNN	**0.87**	1.57	**1.70%**	1.33	2.80	2.80%	1.65	3.75	3.70%	1.95	4.49	4.56%
AGCRN	0.95	1.81	1.94%	1.37	2.92	2.94%	1.69	3.87	3.82%	1.99	4.61	4.62%
GTS	0.91	1.64	1.77%	1.32	2.80	2.75%	1.63	3.74	**3.63%**	**1.90**	**4.40**	**4.44%**
GRU	1.29	2.46	2.54%	1.89	3.53	3.98%	2.27	4.24	5.02%	2.65	4.90	5.92%
GRU-I	1.30	2.57	2.57%	1.89	3.52	3.99%	2.26	4.22	4.99%	2.62	4.89	5.87%
GRU-D	5.40	9.25	13.83%	5.34	9.25	13.76%	5.42	9.26	13.85%	5.41	9.27	13.85%
LSTM-I	1.71	2.69	2.80%	1.97	3.45	4.08%	2.57	5.52	5.62%	2.74	5.00	6.21%
LSTM-M	1.35	2.31	2.71%	1.87	3.39	3.95%	2.33	4.33	5.17%	3.45	8.32	7.29%
SGMN	0.98	1.85	1.88%	1.63	3.40	3.32%	2.29	4.91	4.88%	3.31	6.86	7.32%
GCN-M (ours)	0.91	1.57	1.75%	1.33	**2.72**	2.76%	**1.62**	**3.64**	3.64%	1.95	**4.40**	4.61%

Table 5.4: Performance comparison on **complete** METR-LA dataset

METR-LA	Horizon=1 (5 mins)			Horizon=3 (15 mins)			Horizon=6 (30 mins)			Horizon=12 (60 mins)		
Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	2.45	4.21	5.99%	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
STGCN	2.58	4.32	6.22%	3.04	5.48	8.00%	3.60	6.51	9.97%	4.21	7.37	11.61%
GraphWaveNet	2.41	4.29	5.93%	**2.68**	**5.14**	6.87%	3.06	6.14	8.23%	3.52	7.25	**9.77%**
MTGNN	**2.24**	3.92	**5.39%**	**2.68**	5.16	**6.86%**	**3.05**	6.16	8.19%	**3.50**	**7.24**	9.83%
AGCRN	2.41	4.27	6.08%	2.86	5.54	7.66%	3.22	6.55	8.92%	3.58	7.45	10.24%
GTS	2.32	4.15	6.12%	2.72	5.42	7.11%	3.11	6.47	**7.49%**	3.52	7.49	10.07%
GRU	2.83	4.56	6.78%	3.48	5.80	9.02%	3.97	6.74	10.72%	4.65	7.86	13.00%
GRU-I	2.80	4.52	6.70%	3.49	5.83	9.05%	3.97	6.74	10.75%	4.60	7.88	12.80%
GRU-D	7.46	11.82	24.55%	7.43	11.85	24.62%	7.45	11.84	24.62%	7.47	11.86	24.68%
LSTM-I	2.86	4.57	6.77%	3.57	5.88	9.05%	4.10	6.85	10.94%	4.78	8.13	13.34%
LSTM-M	3.15	5.58	7.03%	3.46	5.74	8.75%	4.08	6.86	10.89%	4.63	7.83	12.92%
SGMN	3.11	6.02	7.01%	4.23	8.54	9.89%	5.46	10.88	13.01%	7.37	13.78	17.81%
GCN-M (ours)	2.34	**3.89**	5.88%	2.74	5.21	6.94%	3.12	6.18	8.25%	3.54	**7.12**	10.01%

train the models for single-step (horizon=1) and multi-step (horizon =3,6,12) forecasting. We report the evaluation errors on each horizon step. We observe from the results that no model achieved evident better performance than the others. However, the first group of works (e.g., DCRNN) performs better than the one-step processing models, which is not surprising as they incorporate the advanced graph models (e.g., *mix-hop propagation* [276]) and training techniques (e.g., *curriculum learning* [276]) to improve the Spatio-temporal forecasting performance. Among the one-step processing models, surprisingly, GRU-D [21] shows much worse performance than the others, which is probably due to the fact that it has been designed for health care applications, whose data have a more stable status than the dynamic traffic data. LSTM-M [24] and SGMN [23], that are designed for traffic forecasting with missing values, show relative good performance in PEMS-BAY especially on single-step forecasting. However, they did not show a clear advantage to the first group of works. The one-step processing models are generally designed for single-step forecasting; their performance gap with the first group of works becomes larger under a multi-step forecasting setting. Even though GCN-M belongs to the one-step processing models, its performance remains close to the first group of works. Moreover, the advanced graph models and training techniques in recent work [276] can be considered to improve the performance of GCN-M further.

5.5.4 RQ 2: Complex scenarios of missing values

In this section, we demonstrate the power of GCN-M in handling complex scenarios of missing values for the purpose of traffic forecasting.

As mentioned previously in Figure 5.1, there are several scenarios of missing values in real-life traffic datasets (e.g., METR-LA): short-range or long-range missing; partial or entire network missing. The results in Table 5.3 and 5.4 did not show the superiority of GCN-M over other models on the original datasets with a low missing rate. Therefore, to test the model's capability of handling complex missing values, we design three scenarios with various missing rates (10%, 20%, and 40%) and remove the observations from the datasets accordingly. We use $\hat{X} \in \mathcal{R}^{n \times f \times t}$ to represent each of the observations to be removed from $\mathcal{X} \in \mathcal{R}^{N \times F \times \tau}$. Then, we design the scenarios of:

- Short-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t = 1$
- Long-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t = \tau$
- Mix-range missing: we randomly set $n \in [1, ..., N]$, $f = F$, $t \in [1, ..., \tau]$

In Table 5.5 and 5.6, we show the performance comparison on the PEMS-BAY and METR-LA datasets under various missing value scenarios. We highlight the best results among the one-step processing models (underlined values) and all the models (bold values). Globally, GCN-M shows the best performance under all the settings compared to the

Table 5.5: Performance comparison on the **incomplete** PEMS-BAY dataset with various random settings on missing values. The results show a one-hour (12-step) average of the forecasting errors. The underlined values represent the best results among the one-step processing models, the bold values represent the best results among all the models.

		Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Short-range missing	DCRNN	1.76	3.94	3.94%	1.82	3.96	4.01%	1.85	4.26	4.04%
	STGCN	1.82	4.11	4.25%	1.91	4.18	4.41%	1.97	4.33	4.42%
	GraphWaveNet	1.69	3.79	3.81%	1.74	3.75	3.75%	1.79	3.87	3.90%
	MTGNN	**1.58**	**3.42**	**3.33%**	1.72	3.78	3.83%	1.83	4.03	3.94%
	AGCRN	1.65	3.81	3.78%	1.66	3.81	3.79%	1.72	3.96	3.95%
	GTS	1.65	3.86	3.74%	1.65	3.86	3.76%	**1.69**	3.92	**3.86%**
	GRU	2.60	4.64	5.75%	2.67	4.78	5.90%	2.86	5.10	6.37%
	GRU-I	2.29	4.28	5.06%	2.31	4.31	5.09%	2.41	4.47	5.38%
	GRU-D	5.38	9.29	13.84%	5.46	9.36	13.96%	7.20	11.58	16.91%
	LSTM-I	2.35	4.33	5.22%	2.82	6.63	6.05%	3.06	7.47	6.56%
	LSTM-M	2.47	4.55	5.50%	2.56	4.70	5.74%	3.34	7.09	7.68%
	SGMN	2.32	4.96	4.94%	2.34	5.01	5.00%	2.45	5.20	5.23%
	GCN-M (ours)	1.62	3.67	3.60%	**1.63**	**3.73**	**3.68%**	1.75	**3.81**	3.90%
Long-range missing	DCRNN	1.83	4.07	4.22%	1.96	4.22	4.42%	2.07	4.45	4.67%
	STGCN	1.92	4.22	4.42%	2.03	4.37	4.72%	2.14	4.52	4.76%
	GraphWaveNet	1.74	3.96	4.03%	1.87	4.09	4.18%	1.94	4.21	4.33%
	MTGNN	**1.65**	**3.68**	**3.72%**	1.89	4.01	4.17%	2.01	4.42	4.61%
	AGCRN	1.72	3.78	3.94%	1.84	4.11	4.13%	1.90	4.18	4.31%
	GTS	1.68	3.86	3.91%	1.78	4.12	4.97%	1.88	4.17	4.22%
	GRU	2.93	5.12	6.32%	3.06	5.31	6.63%	3.35	5.78	7.03%
	GRU-I	2.52	4.51	5.33%	2.53	4.57	5.73%	2.71	4.82	5.51%
	GRU-D	9.33	14.51	22.31%	9.89	13.94	22.86%	11.07	15.88	23.13%
	LSTM-I	2.65	4.65	5.88%	3.13	6.35	6.82%	3.62	9.53	7.12%
	LSTM-M	3.93	7.25	9.17%	5.45	10.06	13.67%	5.57	10.12	14.59%
	SGMN	8.86	12.57	14.54%	11.45	14.56	18.31%	14.62	17.23	23.13%
	GCN-M (ours)	1.70	3.75	3.74%	**1.73**	**3.88**	**3.92%**	**1.79**	**4.07**	**4.14%**
Mix-range missing	DCRNN	1.81	4.01	4.15%	1.91	4.16	4.31%	2.02	4.36	4.52%
	STGCN	1.85	4.13	4.21%	1.98	4.31	4.56%	2.11	4.43	4.68%
	GraphWaveNet	1.72	3.92	3.96%	1.83	4.06	4.14%	1.89	4.11	4.21%
	MTGNN	1.69	3.77	3.78%	1.86	4.03	4.11%	1.98	4.32	4.44%
	AGCRN	1.67	3.85	3.88%	1.72	3.95	3.99%	1.80	4.10	4.13%
	GTS	1.70	3.96	3.92%	1.75	3.98	3.89%	1.79	4.09	4.09%
	GRU	2.71	4.88	6.03%	2.82	5.08	6.28%	3.05	5.43	6.82%
	GRU-I	2.31	4.30	5.11%	2.34	4.39	5.18%	2.40	4.50	5.37%
	GRU-D	8.90	13.71	20.03%	9.46	14.50	21.04%	10.21	15.19	22.44%
	LSTM-I	2.46	4.51	5.49%	2.75	5.85	6.02%	3.39	9.15	6.88%
	LSTM-M	3.86	7.06	8.93%	5.19	9.71	13.15%	5.27	9.74	13.29%
	SGMN	7.41	10.91	13.47%	9.95	13.49	17.56%	13.10	16.96	22.58%
	GCN-M (ours)	**1.65**	**3.67**	**3.69%**	**1.66**	**3.72**	**3.62%**	**1.69**	**3.79**	**3.83%**

Table 5.6: Performance comparison on the **incomplete** METR-LA dataset with various random settings on missing values. The results show a one-hour (12-step) average of the forecasting errors. The underlined values represent the best results among the one-step processing models, the bold values represent the best results among all the models.

		Missing Rate = 10%			Missing Rate = 20%			Missing Rate = 40%		
	Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Short-range missing	DCRNN	3.31	6.61	9.47%	3.44	6.80	9.57%	3.50	6.90	9.68%
	STGCN	3.53	7.08	9.73%	3.59	7.25	10.25%	3.66	7.45	10.41%
	GraphWaveNet	3.28	6.60	9.11%	3.36	6.74	9.50%	3.45	6.81	9.57%
	MTGNN	**2.98**	**6.03**	**8.35%**	3.19	**6.44**	8.69%	3.26	6.59	9.07%
	AGCRN	3.19	6.47	8.81%	3.24	6.60	9.01%	3.25	6.61	9.19%
	GTS	3.08	6.40	8.59%	**3.14**	6.52	**7.58%**	**3.12**	6.56	**8.61%**
	GRU	4.20	7.09	11.27%	4.27	7.16	11.42%	4.45	7.41	11.94%
	GRU-I	4.02	6.83	10.89%	4.03	6.88	10.83%	4.09	6.91	10.88%
	GRU-D	7.50	11.87	24.69%	7.45	11.86	24.66%	7.53	11.91	24.76%
	LSTM-I	4.12	6.89	11.04%	4.18	6.98	11.10%	4.21	7.08	11.26%
	LSTM-M	4.10	6.92	10.91%	4.15	6.98	11.03%	4.26	7.18	11.44%
	SGMN	5.54	10.93	13.17%	5.61	10.99	13.35%	5.81	11.18	13.83%
	GCN-M (ours)	3.17	6.33	8.72%	3.23	6.47	8.99%	3.26	**6.35**	8.98%
Long-range missing	DCRNN	3.46	6.78	9.62%	3.54	6.96	9.75%	3.62	7.02	9.89%
	STGCN	3.71	7.20	9.91%	3.76	7.39	10.42%	3.88	7.66	10.67%
	GraphWaveNet	3.43	6.64	9.07%	3.57	6.92	9.62%	3.61	7.03	10.71%
	MTGNN	3.19	**6.32**	**8.48%**	3.39	6.85	9.21%	3.50	6.95	9.74%
	AGCRN	3.31	6.54	8.94%	3.33	6.68	8.98%	3.33	6.78	9.45%
	GTS	3.25	6.61	8.93%	3.29	6.74	8.85%	3.46	6.86	9.37%
	GRU	4.37	7.28	12.54%	4.47	7.44	11.72%	4.76	7.81	12.71%
	GRU-I	4.20	6.91	11.78%	4.09	6.97	15.42%	4.21	7.03	11.43%
	GRU-D	7.59	11.94	24.72%	7.82	12.46	25.67%	7.96	12.45	26.12%
	LSTM-I	4.20	7.09	11.08%	4.25	7.12	11.32%	4.36	7.32	11.56%
	LSTM-M	4.53	7.47	11.88%	5.21	8.84	15.34%	6.08	10.02	18.13%
	SGMN	9.47	14.30	20.72%	11.49	16.01	24.55%	13.97	18.24	29.10%
	GCN-M (ours)	**3.18**	6.39	8.71%	**3.23**	**6.56**	**8.78%**	**3.27**	**6.68**	**9.12%**
Mix-range missing	DCRNN	3.33	6.69	9.53%	3.47	6.85	9.64%	3.56	6.95	9.78%
	STGCN	3.56	7.12	9.81%	3.64	7.28	10.33%	3.73	7.51	10.62%
	GraphWaveNet	3.28	6.51	9.02%	3.43	6.78	9.52%	3.51	6.94	9.62%
	MTGNN	**3.04**	**6.18**	**7.84%**	3.14	6.72	9.07%	3.44	6.82	9.12%
	AGCRN	3.19	6.49	8.77%	3.21	6.56	8.95%	3.26	6.65	8.98%
	GTS	3.12	6.51	8.61%	3.22	6.61	8.84%	3.34	6.72	8.86%
	GRU	4.30	7.14	11.47%	4.35	7.31	11.68%	4.65	7.72	12.58%
	GRU-I	4.05	6.83	10.94%	4.01	6.86	10.83%	4.11	6.97	10.98%
	GRU-D	7.53	11.89	24.74%	7.71	12.32	25.43%	7.89	12.34	25.63%
	LSTM-I	4.15	6.94	11.06%	4.19	7.01	11.18%	4.30	7.18	11.35%
	LSTM-M	4.40	7.38	11.91%	5.14	8.77	14.92%	6.02	9.92	17.88%
	SGMN	9.33	14.16	20.47%	11.42	15.87	24.40%	13.84	18.13	28.97%
	GCN-M (ours)	3.08	6.34	8.59%	**3.12**	**6.42**	**8.71%**	**3.23**	**6.50**	**8.76%**

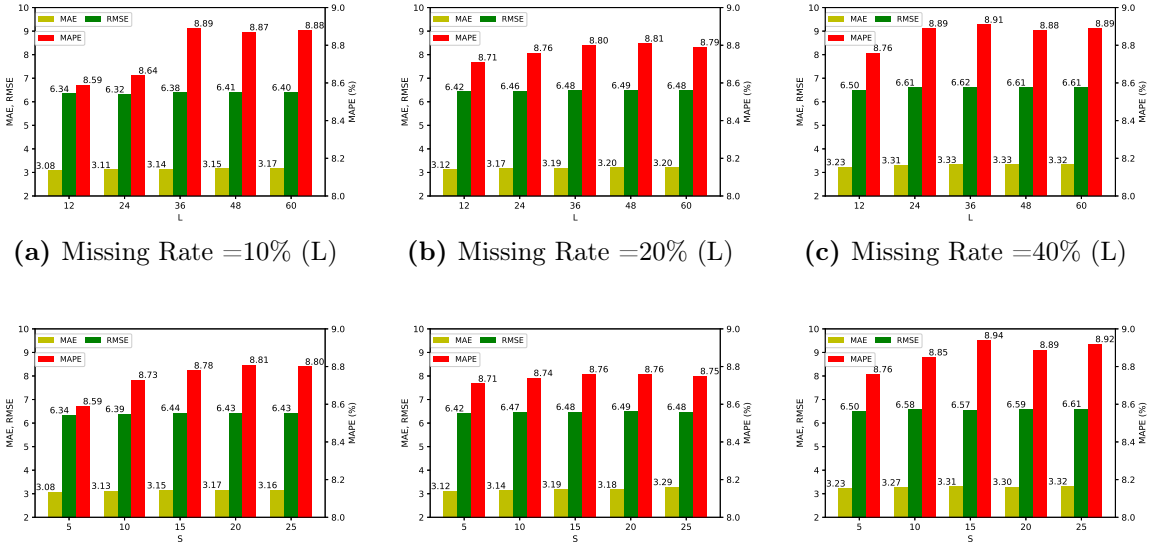
one-step processing models, in which the graph-based model SGMN [23] performs much worse than other one-step processing models under long-range and mix-range missing settings, indicating that it applies only on simple missing scenarios, i.e., short-range random missing. GCN-M does not always show superiority compared with the first group of works, especially in the short-range missing scenario, where MTGNN and GTS usually show good performances. Besides, MTGNN typically performs better than GCN-M when the missing rate is low (10%), except under the mix-range missing scenario of PEMS-BAY. We can draw an interesting conclusion from this observation: a robust Spatio-temporal forecasting model can offset the impact of the missing values to some extent, as it allows exploring the information thoroughly from the observed measures. GCN-M becomes the best forecasting model when the missing rate gets higher, as the missing values become a more critical factor that impacts the forecasting model than Spatio-temporal pattern modeling.

Compared to the short-range missing scenario, GCN-M shows a more robust performance under long-range and mix-range missing scenarios, where the recent temporal values and the nearby nodes' values are not always observed. The multi-scale memory block in GCN-M allows enriching the traffic embedding at each timestamp, thus making the model robust in the two complex scenarios. The memory block searches for the periodic global patterns from historical data and the valuable local features from nearby nodes or recent observations at each timestamp. When nearby nodes values are unobserved, GCN-M favors more on recent observations and vice versa. The memory module with the periodic historical patterns can distinguish the inherent zero values from the missing values, as the zero values usually show periodicity while missing values show contingency [303]. The current node readings combined with the historical patterns will eliminate the effect from missing values but conserve that of zero values.

In Figure 5.6, we show the effects of the memory module's parameters L and S on the model's performance. The two parameters represent the searching range of the local temporal and spatial features, respectively. We report the model's evaluation errors with various missing rates. From the results in Figure 5.6, we observe that when the searching range becomes more extensive, the model's performance decreases more. This can be explained by the fact that the mean value of a larger space and the less recent observations will lead to a weaker information dependency with the current timestamp, thus affecting the information enrichment of the traffic embedding. In real-life datasets, we can set the parameters from a small value, such as considering local features during the last one hour (L=12) with five nearest sensor nodes (S=5).

5.5.5 RQ 3: Dynamic Graph Modeling

In the dynamic traffic system, the spatial dependency can be considered as a dynamic system status, which evolves along time [302]. The traffic observations at each timestamp are always adopted to characterize the dynamic traffic status and help learn the dynamic graphs [277]. However, due to the missed observations, the traffic status at certain times-



(a) Missing Rate =10% (L) **(b)** Missing Rate =20% (L) **(c)** Missing Rate =40% (L)

(d) Missing Rate =10% (S) **(e)** Missing Rate =20% (S) **(f)** Missing Rate =40% (S)

Figure 5.6: Parameters effects: we report the model errors of GCN-M on METR-LA dataset considering *a,b,c) L* observed samples before current timestamp for constructing the *Empirical Temporal Mean* in equation 5.1; *d,e,f) S* observed samples nearby current node for constructing the *Empirical Spatial Mean* in equation 5.2.

tamps can not be characterized, thus affecting the dynamic graph learning process.

This issue can be handled by the enriched traffic embeddings proposed in GCN-M. It allows considering the local static features and global historical patterns to avoid the deviation introduced by the missing values and help learn the dynamic graphs. To validate the performance of the learned dynamic graphs, we designed the following variants of our GCN-M model:

- GCN-M-obs: instead of using the enriched traffic embeddings, the raw traffic observations [277] are adopted to construct dynamic graphs.
- GCN-M-adp: instead of learning dynamic graphs and applying dynamic convolution, an adaptive static graph [278] is learned to do the graph convolution.
- GCN-M-pre: instead of learning graphs from the traffic embedding or observations, the predefined graphs [237] calculated with by the directed distances between traffic nodes are adopted for doing graph convolution.
- GCN-M-com: combine both predefined and learned static graphs [278] to do the graph convolution.

We show in Table 5.7 the performance comparison on various model variants of the spatial graph modeling. We report the model errors on multiple horizons. We consider the complex scenario of mix-range missing values with a missing rate of 40% on both

Table 5.7: Performance comparison on graph module in mix-range missing value scenario with missing rate = 40%

		Horizon=3 (15 mins)			Horizon=6 (30 mins)			Horizon=12 (60 mins)		
	Models	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
PEMS-BAY	GCN-M-obs	1.63	3.48	3.53%	1.93	4.16	4.31%	2.25	4.81	5.10%
	GCN-M-adp	1.53	3.11	3.14%	1.82	3.93	4.03%	2.14	4.72	4.92%
	GCN-M-pre	1.61	3.27	3.21%	1.87	4.05	4.11%	2.18	4.74	5.03%
	GCN-M-com	1.54	3.13	3.11%	1.79	3.92	3.97%	2.11	4.62	3.91%
	GCN-M	**1.45**	**3.03**	**3.09%**	**1.70**	**3.81**	**3.89%**	**2.06**	**4.64**	**4.86%**
METR-LA	GCN-M-obs	2.97	5.68	7.71%	3.31	6.57	8.78%	3.71	7.54	10.07%
	GCN-M-adp	2.84	5.51	7.44%	3.19	6.41	8.59%	3.68	7.34	9.96%
	GCN-M-pre	2.92	5.56	7.64%	3.23	6.42	8.63%	3.72	7.42	10.04%
	GCN-M-com	2.84	5.52	7.45%	3.17	6.4	8.63%	3.68	7.41	9.97%
	GCN-M	**2.82**	**5.47**	**7.42%**	**3.16**	**6.38**	**8.55%**	**3.58**	**7.31**	**9.92%**

PEMS-BAY and METR-LA datasets. The results in Table 5.7 suggest that the dynamic graphs learned from the enriched traffic embeddings perform the best compared to other variants. In contrast, the model obtains the worst performance when learning the dynamic graphs from the raw observations, which is mainly due to the missing values hindering the graph learning process in inferring the dynamic traffic status. GCN-M-obs perform even worse than GCN-M-adp in which the static graph is learned from the entire observations, eliminating the effect from local missing values.

5.5.6 Discussions

Our approach has several advantages. First, starting from the real-world data, GCN-M considered the complex scenarios of missing values in traffic data. Different from the previous work [21, 23, 24] which consider the missing value from a part of the real-life scenarios: either under the short-range or long-range missing settings, under partial or entire networking missing settings. GCN-M considers the complex mix-missing value context covering various real-life scenarios for missing values.

Second, GCN-M is capable of handling such complex missing value scenarios with a multi-scale memory module which combines local Spatio-temporal features (short-range missing, partial and entire network missing) and global historical patterns (long-range missing) to generate the enriched traffic embeddings; the embeddings allow distinguishing the inherent zero values from the missing values. In this way, GCN-M jointly models the Spatio-temporal patterns and missing values in one-step processing, which generally allows a better model performance than two-step processing [23].

Third, GCN-M allows generating reliable dynamic graphs from the enriched traffic embeddings, which opens a path for learning robust dynamic graphs under missing value settings. Moreover, the generated dynamic graphs can corporate with various advanced

graph convolution modules [276] to improve the model's performance further.

Last but not least, even though GCN-M is designed for traffic forecasting, it is applicable to wider application domains sharing similar Spatio-temporal characteristics and missing-value scenarios, such as crowd flow forecasting [19], weather and air pollution forecasting [20], etc. The Spatio-temporal patterns in those data and the missing values caused by the sensor issues or control center errors form similar research problems to our work.

Table 5.8: Model efficiency: training time per epoch (s)

Models	PEMS-BAY	METR-LA	Models	PEMS-BAY	METR-LA
DCRNN	468.22	178.23	GRU	3.65	2.45
STGCN	55.32	27.70	GRU-I	4.22	3.67
GraphWaveNet	118.77	48.16	GRU-D	7.82	5.43
MTGNN	86.20	38.70	LSTM-I	4.32	4.64
AGCRN	67.40	32.9	LSTM-M	8.12	5.76
GTS	191.4	62.3	SGMN	3.45	2.38
GCN-M (ours)	241.69	118.65	-	-	-

However, GCN-M does have a limitation in terms of computational efficiency. Table 5.8 shows the per epoch training time comparison on the full datasets between GCN-M and the baseline models. The one-step processing baseline models are much more efficient than other models. This is basically because of their simple structure without integrating the costly graph convolution modules. GCN-M still performs better than DCRNN, but worse than other forecasting models. This is mainly caused by two factors: 1) generating the enriched traffic embeddings requires a huge computation cost on the attention score's calculation in the memory module; 2) generating the dynamic graphs for graph convolution requires learning a large number of parameters, thus increasing computation cost. Possible solutions might be to reduce the time complexity for calculating the attention with *Prob-Sparse Attention* proposed in [134] and to apply more efficient dynamic graph convolution such as graph tensor decomposition [302].

5.6 Conclusion

In this chapter, we propose GCN-M, a graph convolutional network-based model for handling complex missing values in traffic forecasting. Different from the general multivariate time series (MTS), traffic data is an MTS with geo-locations for each variable (i.e., data source), a.k.a., *geo-located time series*. GCN-M allows modeling the temporal and inter-variable relationships between traffic data under real-world context.

Precisely, we studied the complex scenario where missing traffic values occur on both short & long ranges and on partial & entire transportation network. The enriched traffic embeddings learned by a Spatio-temporal memory module allow handling the complex missing values and constructing dynamic traffic graphs to improve the model's perfor-

mance. A joint model optimization is applied to consider missing values and traffic forecasting in one-step processing. We compare GCN-M with the one-step processing models, which are specifically designed for processing incomplete traffic data and the recent advanced traffic forecasting models. The extensive experiments on two benchmark traffic datasets with 12 baselines demonstrate that GCN-M shows a clear advantage under various scenarios of complex missing values, as compared to the advanced traffic forecasting models, while at the same time it maintains comparable performance on complete traffic datasets. These experiments also provide an up-to-date comparison of the traffic forecasting models would it be with or without missing values. In future work, we will explore the aforementioned optimizations to reduce computational costs. From a longer-term perspective, one can consider noisy data or external events that may impact the predictions.

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

Time series mining is a big research domain with countless real-life applications. The representation learning on time series is a key element for bridging the time series mining and classic machine learning approaches. The time series representation can be divided into three categories: *transformation-based*, *local pattern-based* and *model-based*. Each category is along with various research purpose and concrete research problems with the time series data. From the time series representation, we conduct three interdisciplinary studies on wider time series contexts: (i) representation learning on time series in dynamic streaming context; (ii) representation learning on multivariate time series with label constraint; (iii) representation learning on geo-located time series with imperfect data.

For learning the time series representation in the streaming context, we claim and demonstrate that the Shapelet, a *local pattern-based* representation, is the best choice to answer the question of interpretability in the dynamic streaming context. Different from the learning-based Shapelets [99, 100, 101], we base our proposal on the extraction-based Shapelets [14] which are the shape-based features extracted directly from the original time series and naturally interpretable. Despite being an interpretable feature, the Shapelet can not usually be extracted in an explainable manner. For this reason, we designed SMAP (Shapelet extraction on MAtrix Profile), an explainable extraction process with the help of the *Matrix Profile*, which is integrated into a distributed framework based on Spark to accelerate the extraction process. For the dynamic streaming context, the *Matrix Profile*-based framework is applicable to two different streaming scenarios. First, the time series stream may come continuously within a stable data distribution (i.e., no Concept Drift). This is quite common in real-life applications. For instance, the medical database can be continuously enriched with more patient cases. The learning model needs to be continuously updated without being re-trained from scratch. To this end, we propose ISMAP (Incremental SMAP), which is capable of avoiding the redundant information's

computation by adopting an interleaved *Test-then-Train* strategy [5] with an extra Shapelet evaluation process over input instances. In this manner, ISMAP significantly improves the system's efficiency with an exchange of a negligible decrease of accuracy. Second, the time series stream can come continuously under a dynamic data distribution (i.e., Concept Drift). We adjust the conventional strategies of Concept Drift detection into the context of Time Series Stream, which allows a proactive elimination of the outdated data cached in the memory. The system can be applied in the scenario where an existing dataset should be enriched with new knowledge but without a human loop in the middle.

The lack of annotation is a critical issue in machine learning. It becomes even more challenging when learning from time series (especially multivariate time series). The real-valued sequence is not as interpretable as the classic data (e.g., image, text) for humans. Therefore, the post-labeling on time series is usually more costly than classic data. When considering the label constraint in multivariate time series learning, we should explore at the same time the characteristics of the multivariate time series and the techniques of semi-supervised learning on time series. On the one hand, from univariate time series to multivariate time series, one key challenge is how we can model the interactions between the variables. We designed a simple but powerful module named *Spatial Modeling Block* (SMB), which models the dynamic variable interactions at a local-segment level. SMB shows a higher performance than previous work, which generally considers the fixed sequence-level interactions. This module is parameter-free and can be easily integrated into various advanced frameworks for multivariate time series learning. On the other hand, considering the label constraint when learning from time series, we proposed a semi-supervised model named SMATE based on the auto-encoder structure. SMATE allows learning the Spatial-temporal representation on weakly-labeled multivariate time series. Specifically, a Spatial-temporal encoder maps the temporal dynamic features and the spatial dynamic interactions into a low dimensional embedding space. A semi-supervised three-step regularization process is proposed to compel the model in learning class-separable representation. The weak supervision on the embedding space allows building a reliable classifier, which is extremely valuable in real-life scenarios with label shortage issues. In addition, SMATE allows for visual interpretability in both the learned representation and the semi-supervised representation learning process.

Finally, even though the multivariate time series has a complex structure, the concrete, real-life contexts may introduce auxiliary information on the variables to help model the interactions. For instance, in the Smart City context, the sensor data with spatial information can be represented as geo-located time series. The representation learning on geo-located time series should consider more realistic problems, such as imperfect data with missing values. Meanwhile, we are capable of modeling the variable interactions with the external spatial information. To this end, we conduct the representation learning on the geo-located time series with missing values. Specifically, we consider a popular application of geo-located time series: traffic forecasting. In addition, since the real-world traffic data always comes with data quality issues, such as missing values. To this end, we propose

GCN-M, a graph convolutional network-based model for handling complex missing values in traffic forecasting tasks. Different from the two-step processing approaches, which tackle missing values and forecasting tasks separately, GCN-M allows jointly modeling the Spatio-temporal patterns and the missing values in one-step processing, leading to a better forecasting performance. On the one hand, we studied the complex missing scenario where the values can be missing on both short & long ranges and on partial & entire variables (i.e., transportation network). Specifically, we designed a Spatio-temporal memory module based on the memory network [301] to handle the complex missing values. The memory module first constructs local statistic features from spatial and temporal dimensions to handle short-range missing values. Then, the global historical features are extracted for processing long-range missing blocks. The combined local-global features allow not only identifying the missing measures from the inherent zero values but also enriching the traffic embeddings (i.e., geo-located time series representation). On the other hand, we propose to learn a dynamic graph for modeling the evolving interactions between the spatial nodes. One key advantage of our proposal is that the dynamic graph can be learned from the traffic data with complex missing values. We compare GCN-M with other one-step processing models, which are specifically designed for processing incomplete traffic data and the recent advanced traffic forecasting models. GCN-M shows a clear advantage under various scenarios of complex missing values. Compared to the most advanced traffic forecasting models in the literature, GCN-M maintains comparable performance on complete traffic datasets. Our experiments also provide an up-to-date comparison of the traffic forecasting models would it be with or without missing values.

6.2 Perspectives

Beyond our current work, the representation learning on time series has a wide range of applications. It allows bridging the complex time series analysis and advanced machine learning models in the state of the art. Here, we give some perspectives on our current contributions.

First, in the streaming context, one possible extension is to handle the multivariate time series (MTS) stream. The Shapelet representation in MTS can be either a combination of the Shapelets extracted asynchronously from each variable [223, 98], or a multidimensional Shapelet extracted directly from the MTS, which is rarely studied in the literature. The key question here is how to build the Shapelet representation in MTS via Matrix Profile, which is designed to support the streaming context. Since the Matrix Profile supports only the operations between the univariate time series, a careful design (e.g., pattern combination) for modeling the multivariate patterns is envisaged. Authors in [305] tend to extract the sub-dimensional motif in MTS via Matrix Profile, which is inspiring for us to extract the Shapelets in MTS. The sub-dimensional Shapelet can be a multidimensional subsequence for which only a subset of dimensions is selected. The MTS can then be represented by a (set of) sub-dimensional Shapelet(s) extracted with a k-dimensional Matrix Profile.

Second, there are several possible extensions for our semi-supervised model for multivariate time series representation learning. The first one is to improve further the modeling of the dynamic interactions between the variables. It can be done with the idea in our proposal GCN-M, which applied a dynamic graph neural network to model the spatial interactions between the spatial nodes. The variables in a normal MTS can be considered as a set of spatial nodes with implicit spatial interactions. Recent work [276, 279] learn an implicit graph from the traffic data and show it achieves a comparable or even better performance than that with a predefined graph. In this manner, even without external spatial information, we are capable of learning a dynamic graph to represent the dynamic interactions between the variables. Another possible extension is to improve the performance of the semi-supervised learning framework. It can be done with the time series domain adaptation [297]. The main idea is to pre-train a representation of time series data from other (similar) domains, which learns the shared general time series characteristics. The pre-trained representation can be adapted or fine-tuned to our target domain with the weakly-labeled MTS. In this way, the information in our weakly-labeled MTS can be maximally explored to train the model.

Finally, we aim to improve the efficiency of GCN-M and apply GCN-M to a broader context. The computational efficiency is the main limitation of GCN-M, which is mainly caused by the attention mechanism in the memory module and the dynamic graph convolution on the enriched traffic embeddings. On the one hand, calculating the attention score is costly as it requires computing the dot products of every query-key pair (local Spatio-temporal statistic features and global historical patterns). This issue can be tackled by recent efficient attention mechanisms, such as *ProbSparse Attention* proposed in [134], which allows each key only to attend to a certain number of dominant queries. On the other hand, since the graph convolution is conducted on each of the dynamic graphs and its relevant traffic embedding, the computational cost is multiple times that of learning with a static graph. One possible solution is to adopt the graph tensor decomposition [302], which represents the graphs at all time slots as a tensor. The tensor decomposition allows simplifying the model and exploiting the low-rank characteristic of the dynamic spatial dependencies.

Bibliography

[1] J. Han, J. Pei, and M. Kamber, Data mining: concepts and techniques. Elsevier, 2011.

[2] P. Esling and C. Agon, "Time-series data mining," ACM Computing Surveys (CSUR), vol. 45, no. 1, pp. 1–34, 2012.

[3] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," Data mining and knowledge discovery, vol. 33, no. 4, pp. 917–963, 2019.

[4] B. Lim and S. Zohren, "Time-series forecasting with deep learning: a survey," Philosophical Transactions of the Royal Society A, vol. 379, no. 2194, p. 20200209, 2021.

[5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM computing surveys (CSUR), vol. 46, no. 4, pp. 1–37, 2014.

[6] J. Lines, J. Hills, and A. Bagnall, "The Collective of Transformation-Based Ensembles for Time-Series Classification," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 9, pp. 2522–2535, 2015.

[7] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in 2016 IEEE 16th international conference on data mining (ICDM). IEEE, 2016, pp. 1041–1046.

[8] K. Ueno, A. Xi, E. Keogh, and D. J. Lee, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," Proceedings - IEEE International Conference on Data Mining, ICDM, no. December, pp. 623–632, 2006.

[9] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," Journal of Intelligent Information Systems, vol. 39, no. 2, pp. 287–315, 2012.

[10] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," IEEE Access, vol. 6, pp. 1662–1669, 2017.

[11] J. Wang, Z. Wang, J. Li, and J. Wu, "Multilevel wavelet decomposition network for interpretable time series analysis," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2437–2446.

[12] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: exceptionally fast and accurate time series classification using random convolutional kernels," Data Mining and Knowledge Discovery, vol. 34, no. 5, pp. 1454–1495, 2020.

[13] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in 2016 IEEE 16th international conference on data mining (ICDM). Ieee, 2016, pp. 1317–1322.

[14] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009, pp. 947–956.

[15] K.-W. Chang, B. Deka, W.-M. W. Hwu, and D. Roth, "Efficient pattern-based time series classification on gpu," in 2012 IEEE 12th International Conference on Data Mining. IEEE, 2012, pp. 131–140.

[16] J. Zuo, K. Zeitouni, and Y. Taher, "Exploring interpretable features for large time series with se4tec." in EDBT, 2019, pp. 606–609.

[17] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," Advances in Neural Information Processing Systems, vol. 32, pp. 4650–4661, 2019.

[18] L. Wu, I. En-Hsu, Y. J. Yi, F. Xu, Q. Lei, and M. J. Witbrock, "Random Warping Series: A Random Features Method for Time-Series Embedding," in AISTATS, 2018.

[19] P. Xie, T. Li, J. Liu, S. Du, X. Yang, and J. Zhang, "Urban flow prediction from spatiotemporal data using machine learning: A survey," Information Fusion, vol. 59, pp. 1–12, 2020.

[20] J. Han, H. Liu, H. Zhu, H. Xiong, and D. Dou, "Joint air quality and weather prediction based on multi-adversarial spatiotemporal networks," in Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021.

[21] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," Scientific reports, vol. 8, no. 1, pp. 1–12, 2018.

[22] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values," Transportation Research Part C: Emerging Technologies, vol. 118, p. 102674, 2020.

[23] Z. Cui, L. Lin, Z. Pu, and Y. Wang, "Graph markov network for traffic forecasting with missing data," Transportation Research Part C: Emerging Technologies, vol. 117, p. 102671, 2020.

[24] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "Lstm-based traffic flow prediction with missing data," Neurocomputing, vol. 318, pp. 297–305, 2018.

[25] X. Tang, H. Yao, Y. Sun, C. Aggarwal, P. Mitra, and S. Wang, "Joint modeling of local and global temporal dynamics for multivariate time series forecasting with missing values," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 5956–5963.

[26] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration," Data Mining and knowledge discovery, vol. 7, no. 4, pp. 349–371, 2003.

[27] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[28] R. Wu and E. Keogh, "Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress," IEEE Transactions on Knowledge and Data Engineering, 2021.

[29] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning representations for time series clustering," Advances in neural information processing systems, vol. 32, pp. 3781–3791, 2019.

[30] V. Fortuin, M. Hüser, F. Locatello, H. Strathmann, and G. Rätsch, "Som-vae: Interpretable discrete representation learning on time series," in International Conference on Learning Representations (ICLR 2019). OpenReview. net, 2019.

[31] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," DMKD, 2013.

[32] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in Data mining in time series databases. World Scientific, 2004, pp. 1–21.

[33] H. El Hafyani, K. Zeitouni, Y. Taher, and M. Abboud, "Leveraging change point detection for activity transition mining in the context of environmental crowdsensing," in Actes de la conférence BDA 2020, 2020, p. 64.

[34] D. Hallac, S. Vare, S. Boyd, and J. Leskovec, "Toeplitz inverse covariance-based clustering of multivariate time series data," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 215–223.

[35] H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," in Proceedings of the Twelfth International Conference on Data Engineering. IEEE, 1996, pp. 536–545.

[36] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art," Proceedings of the VLDB Endowment, vol. 12, no. 2, 2020.

[37] P. Patel, E. Keogh, J. Lin, and S. Lonardi, "Mining motifs in massive time series databases," in 2002 IEEE International Conference on Data Mining, 2002. Proceedings. IEEE, 2002, pp. 370–377.

[38] E. A. Maharaj and A. M. Alonso, "Discriminant analysis of multivariate time series: Application to diagnosis based on ecg signals," Computational Statistics & Data Analysis, vol. 70, pp. 67–87, 2014.

[39] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, "Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins," in 2016 IEEE 16th international conference on data mining (ICDM). IEEE, 2016, pp. 739–748.

[40] N. Begum and E. Keogh, "Rare time series motif discovery from unbounded streams," Proceedings of the VLDB Endowment, vol. 8, no. 2, pp. 149–160, 2014.

[41] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 5, 2021, pp. 4027–4035.

[42] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga, "Usad: Unsupervised anomaly detection on multivariate time series," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 3395–3404.

[43] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas, "Coconut: a scalable bottom-up approach for building data series indexes," Proceedings of the VLDB Endowment, vol. 11, no. 6, pp. 677–690, 2018.

[44] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," Data Mining and Knowledge Discovery, vol. 29, no. 6, pp. 1505–1530, 2015.

[45] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," Knowledge and information Systems, vol. 3, no. 3, pp. 263–286, 2001.

[46] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary lp norms," in Proceedings of the 26th International Conference on Very Large Data Bases, 2000, pp. 385–394.

[47] E. J. Keogh and M. J. Pazzani, "A simple dimensionality reduction technique for fast similarity search in large time series databases," in Pacific-Asia conference on knowledge discovery and data mining. Springer, 2000, pp. 122–133.

[48] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," in Proceedings of the 2001 ACM SIGMOD international conference on Management of data, 2001, pp. 151–162.

[49] J. Lin, M. Vlachos, E. Keogh, D. Gunopulos, J. Liu, S. Yu, and J. Le, "A mpaa-based iterative clustering algorithm augmented by nearest neighbors search for time-series data streams," in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2005, pp. 333–342.

[50] S. Lee, D. Kwon, and S. Lee, "Dimensionality reduction for indexing time series based on the minimum distance," Journal of Information Science and Engineering, vol. 19, no. 4, pp. 697–711, 2003.

[51] J. Aßfalg, H.-P. Kriegel, P. Kröger, P. Kunath, A. Pryakhin, and M. Renz, "Similarity search in multimedia time series data using amplitude-level features," in International Conference on Multimedia Modeling. Springer, 2008, pp. 123–133.

[52] C. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi, "A novel bit level time series representation with implication of similarity search and clustering," in Pacific-Asia conference on knowledge discovery and data mining. Springer, 2005, pp. 771–777.

[53] A. Bagnall, C. Ratanamahatana, E. Keogh, S. Lonardi, G. Janacek et al., "A bit level representation for time series data mining with shape based similarity," Data mining and knowledge discovery, vol. 13, no. 1, pp. 11–40, 2006.

[54] D. Tiano, A. Bonifati, and R. Ng, "Feature-driven time series clustering." in EDBT, 2021, pp. 349–354.

[55] ——, "Featts: Feature-based time series clustering," in Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2784–2788.

[56] ——, "Human-centered clustering for time series data," in 3rd Workshop on Data Science with Human in the Loop@ KDD 2021, 2021.

[57] P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying representative trends in massive time series data sets using sketches," in 26th International Conference on Very Large Data Bases, VLDB 2000, 2000, pp. 363–372.

[58] G. Reeves, J. Liu, S. Nath, and F. Zhao, "Managing massive time series streams with multi-scale compressed trickles," Proceedings of the VLDB Endowment, vol. 2, no. 1, pp. 97–108, 2009.

[59] A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 248–257.

[60] C. Bock, M. Togninalli, E. Ghisu, T. Gumbsch, B. Rieck, and K. Borgwardt, "A wasserstein subsequence kernel for time series," in 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 2019, pp. 964–969.

[61] J. Paparrizos and M. J. Franklin, "Grail: efficient time-series representation learning," Proceedings of the VLDB Endowment, vol. 12, no. 11, pp. 1762–1777, 2019.

[62] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in International conference on foundations of data organization and algorithms. Springer, 1993, pp. 69–84.

[63] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," ACM Sigmod Record, vol. 23, no. 2, pp. 419–429, 1994.

[64] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," Acm Sigmod Record, vol. 26, no. 2, pp. 289–300, 1997.

[65] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004, pp. 599–610.

[66] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337). IEEE, 1999, pp. 126–133.

[67] F.-P. Chan, A.-C. Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: with and without time warping," IEEE Transactions on knowledge and data engineering, vol. 15, no. 3, pp. 686–705, 2003.

[68] I. Popivanov and R. J. Miller, "Similarity search over time-series data using wavelets," in Proceedings 18th international conference on data engineering. IEEE, 2002, pp. 212–221.

[69] D. E. Shasha and Y. Zhu, High performance discovery in time series: techniques and case studies. Springer Science & Business Media, 2004.

[70] P. K. Dash, M. Nayak, M. R. Senapati, and I. W. Lee, "Mining for similarities in time series data using wavelet-based feature vectors and neural networks," Engineering Applications of Artificial Intelligence, vol. 20, no. 2, pp. 185–201, 2007.

[71] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "A comparison of dft and dwt based similarity search in time-series databases," in Proceedings of the ninth international conference on Information and knowledge management, 2000, pp. 488–495.

[72] K. Kawagoe and T. Ueda, "A similarity search method of time series data with combination of fourier and wavelet transforms," in Proceedings Ninth International Symposium on Temporal Representation and Reasoning. IEEE, 2002, pp. 86–92.

[73] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos, "Identifying similarities, periodicities and bursts for online search queries," in Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004, pp. 131–142.

[74] Z. R. Struzik and A. Siebes, "Measuring time series similarity through large singular features revealed with wavelet transformation," in Proceedings. Tenth International Workshop on Database and Expert Systems Applications. DEXA 99. IEEE, 1999, pp. 162–166.

[75] V. Megalooikonomou, G. Li, and Q. Wang, "A dimensionality reduction technique for efficient similarity analysis of time series databases," in Proceedings of the thirteenth ACM international conference on Information and knowledge management, 2004, pp. 160–161.

[76] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, "A multiresolution symbolic representation of time series," in 21st International Conference on Data Engineering (ICDE'05). IEEE, 2005, pp. 668–679.

[77] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon, "Similarity preserving representation learning for time series clustering." in IJCAI, vol. 19, 2019, pp. 2845–2851.

[78] E. J. Keogh and M. J. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback." in Kdd, vol. 98, 1998, pp. 239–243.

[79] Y.-W. Huang and P. S. Yu, "Adaptive query processing for time-series data," in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 282–286.

[80] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker, "Landmarks: a new model for similarity-based pattern querying in time series databases," in Proceedings of 16th international conference on data engineering (cat. no. 00cb37073). IEEE, 2000, pp. 33–42.

[81] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu, "Indexable pla for efficient similarity search," in Proceedings of the 33rd international conference on Very large data bases, 2007, pp. 435–446.

[82] T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos, and W. Truppel, "Online amnesic approximation of streaming time series," in Proceedings. 20th International Conference on Data Engineering. IEEE, 2004, pp. 339–349.

[83] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," Data Mining and knowledge discovery, vol. 15, no. 2, pp. 107–144, 2007.

[84] H. André-Jönsson and D. Z. Badal, "Using signature files for querying time-series data," in European Symposium on Principles of Data Mining and Knowledge Discovery. Springer, 1997, pp. 211–220.

[85] C. S. Daw, C. E. A. Finney, and E. R. Tracy, "A review of symbolic analysis of experimental data," Review of Scientific instruments, vol. 74, no. 2, pp. 915–930, 2003.

[86] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03, p. 2, 2003. [Online]. Available: http://portal.acm.org/citation.cfm?doid=882082.882086

[87] B. Lkhagva, Y. Suzuki, and K. Kawagoe, "Extended sax: Extension of symbolic aggregate approximation for financial time series data representation," DEWS2006 4A-i8, vol. 7, 2006.

[88] S. Malinowski, T. Guyet, R. Quiniou, and R. Tavenard, "1d-sax: A novel symbolic representation for time series," in International Symposium on Intelligent Data Analysis. Springer, 2013, pp. 273–284.

[89] Y. Sun, J. Li, J. Liu, B. Sun, and C. Chow, "An improvement of symbolic aggregate approximation distance measure for time series," Neurocomputing, vol. 138, pp. 189–198, 2014.

[90] P. Schäfer and M. Högqvist, "Sfa: a symbolic fourier approximation and index for similarity search in high dimensional datasets," in Proceedings of the 15th international conference on extending database technology, 2012, pp. 516–527.

[91] P. Senin and S. Malinchik, "Sax-vsm: Interpretable time series classification using sax and vector space model," in 2013 IEEE 13th international conference on data mining. IEEE, 2013, pp. 1175–1180.

[92] J. An, H. Chen, K. Furuse, N. Ohbo, and E. Keogh, "Grid-based indexing for large time series databases," in International Conference on Intelligent Data Engineering and Automated Learning. Springer, 2003, pp. 614–621.

[93] A. Bagnall, G. Janacek, and M. Zhang, "Clustering time series from mixture polynomial models with discretised data," 2003.

[94] J. Zakaria, A. Mueen, and E. Keogh, "Clustering time series using unsupervised-shapelets," in 2012 IEEE 12th International Conference on Data Mining. IEEE, 2012, pp. 785–794.

[95] L. Ulanova, N. Begum, and E. Keogh, "Scalable clustering of time series with u-shapelets," in Proceedings of the 2015 SIAM international conference on data mining. SIAM, 2015, pp. 900–908.

[96] X. Wang, J. Lin, P. Senin, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein, "Rpm: Representative pattern mining for efficient time series classification." in EDBT, 2016, pp. 185–196.

[97] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 289–297.

[98] R. Mousheimish, Y. Taher, and K. Zeitouni, "Automatic Learning of Predictive CEP Rules: Bridging the Gap between Data Mining and Complex Event Processing," in DEBS, 2017, p. 158–169.

[99] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 392–401.

[100] Z. Fang, P. Wang, and W. Wang, "Efficient learning interpretable shapelets for accurate time series classification," in 2018 IEEE 34th International Conference on Data Engineering (ICDE). IEEE, 2018, pp. 497–508.

[101] Y. Wang, R. Emonet, E. Fromont, S. Malinowski, and R. Tavenard, "Adversarial regularization for explainable-by-design time series classification," in 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2020, pp. 1079–1087.

[102] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Fast classification of univariate and multivariate time series through shapelet discovery," Knowledge and Information Systems, vol. 49, no. 2, pp. 429–454, 2016.

[103] V. S. S. Fotso, E. M. Nguifo, and P. Vaslin, "Frobenius correlation based u-shapelets discovery for time series clustering," Pattern Recognition, vol. 103, p. 107301, 2020.

[104] J. Grabocka, M. Wistuba, and L. Schmidt-Thieme, "Scalable classification of repetitive time series through frequencies of local polynomials," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 6, pp. 1683–1695, 2014.

[105] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance measures for effective clustering of arima time-series," in Proceedings 2001 IEEE international conference on data mining. IEEE, 2001, pp. 273–280.

[106] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis, "Discovering dynamics using bayesian clustering," in International Symposium on Intelligent Data Analysis. Springer, 1999, pp. 199–209.

[107] A. Panuccio, M. Bicego, and V. Murino, "A hidden markov model-based approach to sequential data clustering," in Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR). Springer, 2002, pp. 734–743.

[108] S. Tonekaboni, D. Eytan, and A. Goldenberg, "Unsupervised representation learning for time series with temporal neighborhood coding," in International Conference on Learning Representations, 2020.

[109] Q. Ma, C. Chen, S. Li, and G. W. Cottrell, "Learning representations for incomplete time series clustering," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 10, 2021, pp. 8837–8846.

[110] D. Song, N. Xia, W. Cheng, H. Chen, and D. Tao, "Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2229–2238.

[111] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series." in CCIA, 2018, pp. 120–129.

[112] A. Hyvarinen and H. Morioka, "Unsupervised feature extraction by time-contrastive learning and nonlinear ica," Advances in Neural Information Processing Systems, vol. 29, 2016.

[113] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff, "Timenet: Pretrained deep recurrent neural network for time series classification," arXiv preprint arXiv:1706.08838, 2017.

[114] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The bulletin of mathematical biophysics, vol. 5, no. 4, pp. 115–133, 1943.

[115] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," Data Mining and Knowledge Discovery, vol. 31, no. 3, pp. 606–660, 2017.

[116] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances," Data Mining and Knowledge Discovery, vol. 35, no. 2, pp. 401–449, 2021.

[117] C. W. Tan, C. Bergmeir, F. Petitjean, and G. I. Webb, "Time series extrinsic regression," Data Mining and Knowledge Discovery, vol. 35, no. 3, pp. 1032–1060, 2021.

[118] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016, pp. 770–778.

[119] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in IJCNN, 2017.

[120] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in Thirty-first AAAI conference on artificial intelligence, 2017.

[121] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," Data Mining and Knowledge Discovery, vol. 34, no. 6, pp. 1936–1962, 2020.

[122] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series classification," in 2019 International Joint Conference on Neural Networks (IJCNN). IEEE, 2019, pp. 1–8.

[123] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, and M. Blumenstein, "Rethinking 1d-cnn for time series classification: A stronger baseline," arXiv preprint arXiv:2002.10061, 2020.

[124] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[125] Y. Hao and H. Cao, "A New Attention Mechanism to Classify Multivariate Time Series," in IJCAI, 2020, pp. 1999–2005.

[126] A. Nanopoulos, R. Alcock, and Y. Manolopoulos, "Feature-based classification of time-series data," International Journal of Computer Research, vol. 10, no. 3, pp. 49–61, 2001.

[127] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," arXiv preprint arXiv:1603.06995, 2016.

[128] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[129] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in CVPR, 2017, pp. 156–165.

[130] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in IJCAI, 2018.

[131] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in IJCAI, 2017.

[132] D. Xu, W. Cheng, B. Zong, D. Song, J. Ni, W. Yu, Y. Liu, H. Chen, and X. Zhang, "Tensorized lstm with adaptive shared memory for learning trends in multivariate time series," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 02, 2020, pp. 1395–1402.

[133] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," International Journal of Forecasting, vol. 37, no. 4, pp. 1748–1764, 2021.

[134] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in Proceedings of AAAI, 2021.

[135] W. Yu, I. Y. Kim, and C. Mechefske, "Analysis of different rnn autoencoder variants for time series classification and machine prognostics," Mechanical Systems and Signal Processing, vol. 149, p. 107322, 2021.

[136] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," Advances in neural information processing systems, vol. 27, 2014.

[137] A. Sagheer and M. Kotb, "Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems," Scientific reports, vol. 9, no. 1, pp. 1–16, 2019.

[138] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2114–2124.

[139] A. Van Den Oord, O. Vinyals et al., "Neural discrete representation learning," Advances in neural information processing systems, vol. 30, 2017.

[140] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological cybernetics, vol. 43, no. 1, pp. 59–69, 1982.

[141] C. J. Geyer, "Practical markov chain monte carlo," Statistical science, pp. 473–483, 1992.

[142] O. Mogren, "C-rnn-gan: Continuous recurrent neural networks with adversarial training," arXiv preprint arXiv:1611.09904, 2016.

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

[144] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," Advances in Neural Information Processing Systems, vol. 32, 2019.

[145] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in International Conference on Artificial Neural Networks. Springer, 2019, pp. 703–716.

[146] K. Nikolaidis, S. Kristiansen, V. Goebel, T. Plagemann, K. Liestøl, and M. Kankanhalli, "Augmenting physiological time series data: A case study for sleep apnea detection," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2019, pp. 376–399.

[147] Y. Luo, X. Cai, Y. Zhang, J. Xu et al., "Multivariate time series imputation with generative adversarial networks," Advances in neural information processing systems, vol. 31, 2018.

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

[149] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," Advances in neural information processing systems, vol. 29, 2016.

[150] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," Advances in neural information processing systems, vol. 30, 2017.

[151] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784, 2014.

[152] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," arXiv preprint arXiv:1511.05644, 2015.

[153] N. Saunshi, O. Plevrakis, S. Arora, M. Khodak, and H. Khandeparkar, "A theoretical analysis of contrastive unsupervised representation learning," in International Conference on Machine Learning. PMLR, 2019, pp. 5628–5637.

[154] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang, "Self-supervised learning: Generative or contrastive," IEEE Transactions on Knowledge and Data Engineering, 2021.

[155] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Advances in neural information processing systems, vol. 26, 2013.

[156] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K.-P. Chun, and G. L. Wong, "Shapenet: A shapelet-neural network approach for multivariate time series classification," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 9, 2021, pp. 8375–8383.

[157] Q. Ma, W. Zhuang, S. Li, D. Huang, and G. Cottrell, "Adversarial dynamic shapelet networks," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 5069–5076.

[158] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, 2016.

[159] M. M. Masud, Q. Chen, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Classification and novel class detection of data streams in a dynamic feature space," in Joint European conference on machine learning and knowledge discovery in databases. Springer, 2010, pp. 337–352.

[160] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in 2016 IEEE 32nd International Conference on Data Engineering (ICDE). IEEE, 2016, pp. 481–492.

[161] J. Zuo, K. Zeitouni, and Y. Taher, "Time series meet data streams: Perspectives of the interdisciplinary collision and applications," in PhD Symposium, Actes de la conférence BDA 2019, 2019.

[162] K. Ueno, X. Xi, E. Keogh, and D.-J. Lee, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," in Sixth International Conference on Data Mining (ICDM'06). IEEE, 2006, pp. 623–632.

[163] S. Kasetty, C. Stafford, G. P. Walker, X. Wang, and E. Keogh, "Real-time classification of streaming sensor data," in 2008 20th IEEE International Conference on Tools with Artificial Intelligence, vol. 1. IEEE, 2008, pp. 149–156.

[164] J. Shieh and E. Keogh, "Polishing the right apple: Anytime classification also benefits data streams with constant arrival times," in 2010 IEEE International Conference on Data Mining. IEEE, 2010, pp. 461–470.

[165] A. Marascu, S. A. Khan, and T. Palpanas, "Scalable similarity matching in streaming time series," in Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 2012, pp. 218–230.

[166] L. Liu, Y. Peng, S. Wang, M. Liu, and Z. Huang, "Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors," Information Sciences, vol. 340, pp. 41–57, 2016.

[167] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 262–270.

[168] H. Hamooni and A. Mueen, "Dispatch: Distributed pattern matching over streaming time series," in 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018, pp. 2890–2899.

[169] Z. Zimmerman, N. S. Senobari, G. Funning, E. Papalexakis, S. Oymak, P. Brisk, and E. Keogh, "Matrix profile xviii: time series mining in the face of fast moving streams using a learned approximate matrix profile," in 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 2019, pp. 936–945.

[170] A. Mueen and E. Keogh, "Online discovery and maintenance of time series motifs," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 1089–1098.

[171] Y. Hao, Y. Chen, J. Zakaria, B. Hu, T. Rakthanmanon, and E. Keogh, "Towards never-ending learning from time series streams," in Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 874–882.

[172] J. Michael, "Where's the evidence that active learning works?" Advances in physiology education, 2006.

[173] L. Wei and E. Keogh, "Semi-supervised time series classification," in KDD, 2006, p. 748–753.

[174] Y. Chen, B. Hu, E. Keogh, and G. E. Batista, "DTW-D: Time Series Semi-Supervised Learning from a Single Example," in KDD, 2013, p. 383–391.

[175] M. N. Nguyen, X. L. Li, and S. K. Ng, "Positive unlabeled learning for time series classification," in IJCAI, 2011, p. 1421–1426.

[176] C. A. Ratanamahatana and D. Wanichsan, "Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification," Soft. Eng., Arti. Intel., Net. & Para./Distri. Comp., pp. 1–14, 2008.

[177] H. Wang, Q. Zhang, J. Wu, S. Pan, and Y. Chen, "Time series feature learning with labeled and unlabeled data," Pattern Recognition, vol. 89, pp. 55–66, 2019.

[178] S. Jawed, J. Grabocka, and L. Schmidt-Thieme, "Self-supervised learning for semi-supervised time series classification," Advances in Knowledge Discovery and Data Mining (PAKDD), vol. 12084, p. 499, 2020.

[179] M. Shokoohi-Yekta, J. Wang, and E. Keogh, "On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case," in SDM, 2015.

[180] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, and E. Keogh, "Generalizing DTW to the multi-dimensional case requires an adaptive approach HHS Public Access," Data Min Knowl Discov, vol. 31, no. 1, pp. 1–31, 2017.

[181] 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, and G. Batista, "The ucr time series classification archive," October 2018, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[182] A. B. Hoang, A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, "The UEA multivariate time series classification archive," Tech. Rep., 2018.

[183] Z. Xu and K. Funaya, "Time series analysis with graph-based semi-supervised learning," in 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA). IEEE, 2015, pp. 1–6.

[184] G. Wilson, J. R. Doppa, and D. J. Cook, "Multi-source deep domain adaptation with weak supervision for time-series sensor data," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1768–1778.

[185] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Transactions on knowledge and data engineering, vol. 22, no. 10, pp. 1345–1359, 2009.

[186] C. Doersch and A. Zisserman, "Multi-task self-supervised visual learning," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2051–2060.

[187] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 6845–6852.

[188] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," Data Mining and Knowledge Discovery, vol. 29, no. 3, pp. 565–592, 2015.

[189] T. Górecki and M. Łuczak, "Using derivatives in time series classification," Data Mining and Knowledge Discovery, vol. 26, no. 2, pp. 310–331, 2013.

[190] T. Górecki and M. Łuczak, "Non-isometric transforms in time series classification using dtw," Know.-Based Syst., vol. 61, pp. 98–108, 2014.

[191] C. A. Ratanamahatana and E. Keogh, "Three Myths about Dynamic Time Warping Data Mining," in Proceedings of the 2005 SIAM International Conference on Data Mining, 2005, pp. 506–510.

[192] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," Information Sciences, vol. 239, pp. 142–153, 2013.

[193] M. G. Baydogan, G. Runger, and E. Tuv, "A bag-of-features framework to classify time series," IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 11, pp. 2796–2802, 2013.

[194] M. Gokce Baydogan, G. Runger, and E. B. Keogh Mustafa Gokce Baydogan, "Time series representation and similarity based on local autopatterns," Data Mining and Knowledge Discovery, vol. 30, pp. 476–509, 2016. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2Fs10618-015-0425-y.pdf

[195] A. Shifaz, C. Pelletier, F. Petitjean, and G. I. Webb, "Ts-chief: a scalable and accurate forest algorithm for time series classification," Data Mining and Knowledge Discovery, vol. 34, no. 3, pp. 742–775, 2020.

[196] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in KDD workshop, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[197] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," Knowledge and information systems, vol. 7, no. 3, pp. 358–386, 2005.

[198] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," Intelligent Data Analysis, vol. 11, no. 5, pp. 561–580, 2007.

[199] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in Proceedings of the 2001 SIAM international conference on data mining. SIAM, 2001, pp. 1–11.

[200] G. G. Yen and K.-C. Lin, "Wavelet packet feature extraction for vibration monitoring," IEEE transactions on industrial electronics, vol. 47, no. 3, pp. 650–667, 2000.

[201] P. Geurts, "Pattern extraction for time series classification," in European conference on principles of data mining and knowledge discovery. Springer, 2001, pp. 115–127.

[202] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 12, pp. 3026–3037, 2014.

[203] I. Nun, P. Protopapas, B. Sim, M. Zhu, R. Dave, N. Castro, and K. Pichara, "Fats: Feature analysis for time series," arXiv preprint arXiv:1506.00010, 2015.

[204] T. Hastie, R. Tibshirani, and J. H. Friedman, The elements of statistical learning: data mining, inference, and prediction. Springer, 2009, vol. 2.

[205] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "Lb_keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures," in Proceedings of the 32nd international conference on Very large data bases. Citeseer, 2006, pp. 882–893.

[206] M. F. Mbouopda and E. M. Nguifo, "Uncertain time series classification with shapelet transform," in 2020 International Conference on Data Mining Workshops (ICDMW). IEEE, 2020, pp. 259–266.

[207] J. Zuo, K. Zeitouni, and Y. Taher, "Incremental and adaptive feature exploration over time series stream," in 2019 IEEE International Conference on Big Data (Big Data). IEEE, 2019, pp. 593–602.

[208] J. Lines and A. Bagnall, "Alternative quality measures for time series shapelets," in International Conference on Intelligent Data Engineering and Automated Learning. Springer, 2012, pp. 475–483.

[209] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in proceedings of the 2013 SIAM International Conference on Data Mining. SIAM, 2013, pp. 668–676.

[210] J. Buhler and M. Tompa, "Finding motifs using random projections," Journal of computational biology, vol. 9, no. 2, pp. 225–242, 2002.

[211] A. Yamaguchi, S. Maya, and K. Ueno, "Rlts: Robust learning time-series shapelets," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2020, pp. 595–611.

[212] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," arXiv preprint arXiv:1712.04621, 2017.

[213] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: a comprehensive review," ACM Computing Surveys (CSUR), vol. 54, no. 3, pp. 1–40, 2021.

[214] O. Stephen, M. Sain, U. J. Maduh, and D.-U. Jeong, "An efficient deep learning approach to pneumonia classification in healthcare," Journal of healthcare engineering, vol. 2019, 2019.

[215] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," Data mining and knowledge Discovery, vol. 13, no. 3, pp. 335–364, 2006.

[216] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall, "The temporal dictionary ensemble (tde) classifier for time series classification," in Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2020, pp. 660–676.

[217] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable dictionary classifiers for time series classification," in International Conference on Intelligent Data Engineering and Automated Learning. Springer, 2019, pp. 11–19.

[218] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "On time series classification with dictionary-based classifiers," Intelligent Data Analysis, vol. 23, no. 5, pp. 1073–1089, 2019.

[219] P. Schäfer and U. Leser, "Fast and Accurate Time Series Classification with WEASEL," in CIKM, 2017.

[220] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, "On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (hive-cote v1. 0)," in International Workshop on Advanced Analytics and Learning on Temporal Data. Springer, 2020, pp. 3–18.

[221] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classification," Machine Learning, vol. 110, no. 11, pp. 3211–3243, 2021.

[222] S. Li, Y. Li, and Y. Fu, "Multi-View Time Series Classification: A Discriminative Bilinear Projection Approach," in CIKM, 2016.

[223] M. S. Cetin, A. Mueen, and V. D. Calhoun, "Shapelet ensemble for multi-dimensional time series," in SDM, 2015, pp. 307–315.

[224] M. Gokce Baydogan, G. Runger, M. G. Baydogan, and G. Runger, "Learning a symbolic representation for multivariate time series classification," Data Min Knowl Disc, vol. 29, pp. 400–422, 2015.

[225] P. Schäfer and U. Leser, "Multivariate Time Series Classification with WEASEL+MUSE," in AALTD, 2018.

[226] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in WAIM, 2014, pp. 298–310.

[227] J. Yang, M. N. Nguyen, P. P. San, X. L. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition," in IJCAI, 2015, p. 3995–4001.

[228] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the Dots: Multivariate Time Series Forecasting with Graph Neural Networks," in KDD, 2020.

[229] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," in ICLR, 2018.

[230] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," Neural Networks, vol. 116, 2019.

[231] Z. Ouyang, P. Ravier, and M. Jabloun, "Stl decomposition of time series can benefit forecasting done by statistical methods but not by machine learning ones," Engineering Proceedings, vol. 5, no. 1, p. 42, 2021.

[232] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," Renewable and Sustainable Energy Reviews, vol. 74, pp. 902–924, 2017.

[233] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," Applied soft computing, vol. 90, p. 106181, 2020.

[234] Z.-L. Sun, T.-M. Choi, K.-F. Au, and Y. Yu, "Sales forecasting using extreme learning machine with applications in fashion retailing," Decision Support Systems, vol. 46, no. 1, pp. 411–419, 2008.

[235] S. D. Campbell and F. X. Diebold, "Weather forecasting for weather derivatives," Journal of the American Statistical Association, vol. 100, no. 469, pp. 6–16, 2005.

[236] X. Yi, J. Zhang, Z. Wang, T. Li, and Y. Zheng, "Deep distributed fusion network for air quality prediction," in Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 965–973.

[237] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in ICLR, 2018.

[238] R. J. Hyndman and G. Athanasopoulos, Forecasting: principles and practice. OTexts, 2018.

[239] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," Econometric reviews, vol. 29, no. 5-6, pp. 594–621, 2010.

[240] T. G. Smith et al., "pmdarima: Arima estimators for Python," 2017–. [Online]. Available: http://www.alkaline-ml.com/pmdarima

[241] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with python," in 9th Python in Science Conference, 2010.

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

[243] R. Taylor, "PyFlux: An open source time series library for python," 2016–. [Online]. Available: https://pyflux.readthedocs.io/en/latest/index.html

[244] H. Lütkepohl and M. Krätzig, Applied time series econometrics. Cambridge university press, 2004.

[245] J. Durbin and S. J. Koopman, "A simple and efficient simulation smoother for state space time series analysis," Biometrika, vol. 89, no. 3, pp. 603–616, 2002.

[246] D. Creal, S. J. Koopman, and A. Lucas, "Generalized autoregressive score models with applications," Journal of Applied Econometrics, vol. 28, no. 5, pp. 777–795, 2013.

[247] A. C. Harvey, Dynamic models for volatility and heavy tails: with applications to financial and economic time series. Cambridge University Press, 2013, vol. 52.

[248] J. Herzen, F. Lässig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. Kościsz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch, "Darts: User-friendly modern machine learning for time series," 2021.

[249] S. J. Taylor and B. Letham, "Forecasting at scale," The American Statistician, vol. 72, no. 1, pp. 37–45, 2018.

[250] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," arXiv preprint arXiv:2101.11174, 2021.

[251] A. Koesdwiady, R. Soua, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: A deep learning approach," IEEE Transactions on Vehicular Technology, vol. 65, no. 12, pp. 9508–9517, 2016.

[252] X. Zhang, C. Huang, Y. Xu, L. Xia, P. Dai, L. Bo, J. Zhang, and Y. Zheng, "Traffic flow forecasting with spatial-temporal graph diffusion network," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 17, 2021, pp. 15 008–15 015.

[253] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1720–1730.

[254] L. A. Cruz, K. Zeitouni, and J. A. F. de Macedo, "Trajectory prediction from a mass of sparse and missing external sensor data," in 2019 20th IEEE International Conference on Mobile Data Management (MDM). IEEE, 2019, pp. 310–319.

[255] L. A. Cruz, K. Zeitouni, T. L. C. da Silva, J. A. F. de Macedo, and J. S. d. Silva, "Location prediction: a deep spatiotemporal learning from external sensors data," Distributed and Parallel Databases, vol. 39, no. 1, pp. 259–280, 2021.

[256] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," Advances in neural information processing systems, vol. 28, 2015.

[257] Y. Wang, M. Long, J. Wang, Z. Gao, and P. S. Yu, "Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms," Advances in neural information processing systems, vol. 30, 2017.

[258] S. Oprea, P. Martinez-Gonzalez, A. Garcia-Garcia, J. A. Castro-Vargas, S. Orts-Escolano, J. Garcia-Rodriguez, and A. Argyros, "A review on deep learning techniques for video prediction," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

[259] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE transactions on neural networks, vol. 20, no. 1, pp. 61–80, 2008.

[260] S.-Y. Shih, F.-K. Sun, and H.-y. Lee, "Temporal pattern attention for multivariate time series forecasting," Machine Learning, vol. 108, no. 8, pp. 1421–1441, 2019.

[261] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in Thirty-first AAAI conference on artificial intelligence, 2017.

[262] X. Shi, Z. Gao, L. Lausen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Deep learning for precipitation nowcasting: A benchmark and a new model," Advances in neural information processing systems, vol. 30, 2017.

[263] Z. Lin, J. Feng, Z. Lu, Y. Li, and D. Jin, "Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis," in Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01, 2019, pp. 1020–1027.

[264] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1, 2018.

[265] Y. Wang, Z. Gao, M. Long, J. Wang, and S. Y. Philip, "Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning," in International Conference on Machine Learning. PMLR, 2018, pp. 5123–5132.

[266] Y. Wang, L. Jiang, M.-H. Yang, L.-J. Li, M. Long, and L. Fei-Fei, "Eidetic 3d lstm: A model for video prediction and beyond," in International conference on learning representations, 2018.

[267] Y. Wang, J. Zhang, H. Zhu, M. Long, J. Wang, and P. S. Yu, "Memory in memory: A predictive neural network for learning higher-order non-stationarity from spatiotemporal dynamics," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9154–9162.

[268] Y.-a. Geng, Q. Li, T. Lin, L. Jiang, L. Xu, D. Zheng, W. Yao, W. Lyu, and Y. Zhang, "Lightnet: A dual spatiotemporal encoder network model for lightning prediction," in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2439–2447.

[269] Z. Lin, M. Li, Z. Zheng, Y. Cheng, and C. Yuan, "Self-attention convlstm for spatiotemporal prediction," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, 2020, pp. 11 531–11 538.

[270] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01, 2019, pp. 922–929.

[271] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, 2020, pp. 914–921.

[272] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," in Proceedings of the AAAI conference on artificial intelligence, vol. 35, no. 5, 2021, pp. 4189–4196.

[273] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, 2020, pp. 1234–1241.

[274] C. Park, C. Lee, H. Bahng, Y. Tae, S. Jin, K. Kim, S. Ko, and J. Choo, "St-grat: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed," in Proceedings of the 29th ACM International conference on information & knowledge management, 2020, pp. 1215–1224.

[275] X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia, and J. Yu, "Traffic flow prediction via spatial temporal graph neural network," in Proceedings of The Web Conference 2020, 2020, pp. 1082–1092.

[276] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in KDD, 2020, pp. 753–763.

[277] F. Li, J. Feng, H. Yan, G. Jin, D. Jin, and Y. Li, "Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution," arXiv preprint arXiv:2104.14917, 2021.

[278] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling." in IJCAI, 2019.

[279] L. BAI, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," Advances in Neural Information Processing Systems, vol. 33, 2020.

[280] C. Shang, J. Chen, and J. Bi, "Discrete graph structure learning for forecasting multiple time series," in International Conference on Learning Representations, 2020.

[281] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting," IEEE Transactions on Knowledge and Data Engineering, 2021.

[282] P. Schäfer, "The BOSS is concerned with time series classification in the presence of noise," Data Min Knowl Disc, vol. 29, pp. 1505–1530, 2015.

[283] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in Proceedings of the First Workshop on Applications of Pattern Analysis. PMLR, 2010, pp. 44–50.

[284] A. Bostrom and A. Bagnall, "Binary shapelet transform for multiclass time series classification," in Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXII. Springer, 2017, pp. 24–46.

[285] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh, "Matrix profile x: Valmod-scalable discovery of variable-length motifs in data series," in Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 1053–1066.

[286] J. Zuo, K. Zeitouni, and Y. Taher, "ISETS: Incremental Shapelet Extraction from Time Series Stream," ECML-PKDD, pp. 790–793, 2019.

[287] S. Gharghabi, S. Imani, A. Bagnall, A. Darvishzadeh, and E. Keogh, "An ultra-fast time series distance measure to allow data mining in more complex real-world deployments," Data Mining and Knowledge Discovery, vol. 34, pp. 1104–1135, 2020.

[288] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, and E. Keogh, "The fastest similarity search algorithm for time series subsequences under euclidean distance," August 2017, http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[289] J. Lines, S. Taylor, and A. Bagnall, "HIVE-COTE: The Hierarchical Vote Collective of Transformation-based Ensembles for Time Series Classification," in IEEE ICDM, 2016.

[290] A. Dorle, F. Li, W. Song, and S. Li, "Learning Discriminative Virtual Sequences for Time Series Classification," in CIKM, 2020, p. 2001–2004.

[291] Y. Bai, L. Wang, Z. Tao, S. Li, and Y. Fu, "Correlative Channel-Aware Fusion for Multi-View Time Series Classification," in AAAI, 2021.

[292] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE TPAMI, pp. 1798–1828, 2013.

[293] Q. Ma, J. Zheng, S. Li, and G. W. Cottrell, "Learning Representations for Time Series Clustering," in NeurIPS, 2019.

[294] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in International conference on machine learning. PMLR, 2015, pp. 448–456.

[295] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in ICML, 2010.

[296] D. P. Kingma and J. Lei Ba, "Adam: A Method for Stochastic Optimization," in ICLR, 2015.

[297] G. Wilson, J. R. Doppa, and D. J. Cook, "Multi-source deep domain adaptation with weak supervision for time-series sensor data," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1768–1778.

[298] J. Zuo, K. Zeitouni, and Y. Taher, "Smate: Semi-supervised spatio-temporal representation learning on multivariate time series," in 2021 IEEE International Conference on Data Mining (ICDM). IEEE, 2021, pp. 1565–1570.

[299] R.-G. Cirstea, B. Yang, and C. Guo, "Graph attention recurrent neural networks for correlated time series forecasting," MileTS19@ KDD, 2019.

[300] J. Yoon, D. Jarrett, and M. Van Der Schaar, "Time-series Generative Adversarial Networks," in NeurIPS, 2019.

[301] J. Weston, S. Chopra, and A. Bordes, "Memory networks," arXiv preprint arXiv:1410.3916, 2014.

[302] L. Han, B. Du, L. Sun, Y. Fu, Y. Lv, and H. Xiong, "Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting," in KDD, 2021, pp. 547–555.

[303] "An introduction to the caltrans performance measurement system (pems)," https://pems.dot.ca.gov/PeMS_Intro_User_Guide_v5.pdf, 2015.

[304] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in NIPS 2014 Workshop on Deep Learning, 2014.

[305] C.-C. M. Yeh, N. Kavantzas, and E. Keogh, "Matrix profile vi: Meaningful multidimensional motif discovery," in 2017 IEEE international conference on data mining (ICDM). IEEE, 2017, pp. 565–574.