

UNIVERSITY OF BAHRAIN

College of Information Technology



New Dimension Reduction Methods Based on Auto-Encoder for Efficient Clustering

A Thesis Submitted in Partial Fulfilment of the Requirements for the
Doctor of Philosophy Degree in Computing and Information Sciences

Submitted By

Hasna AlSaeed

20011935

Supervised By

Prof. Riadh Ksantini

(Associate Professor)

Co-Supervisor

Prof. Faisal AlKhateeb

(Professor)

University of Bahrain

October 2025



Dissertation Deposit Form

Author Name:	Hasna Jaffer Alsaeed		
Email:	20011935@stu.uob.edu.bh	Contact No:	39233020
College:	Information Technology		
Department:	Computer Science		
Thesis Title:	New Dimension Reduction Methods Based on Auto-Encoder for Efficient Clustering		
Year:	2025		

Author's Declaration: I agree to the following conditions:

The above thesis will be available (in the University of Bahrain library and externally) and reproduced as necessary at the discretion of the University of Bahrain. It may also be digitized by the University of Bahrain and made available via the university's repository or via other parties on the Internet.

This condition shall apply to ALL copies including electronic copies.

The above thesis has been provided on the understanding that it is copyright protected material and that no quotation from this thesis may be published without proper acknowledgement.

Signature of the Author:	Hasna Alsaeed	Date:	28-Oct-25
--------------------------	---------------	-------	-----------

DECLARATION

I declare that this work is the result of my own investigation and that it has not already been accepted in substance for any degree, nor is it currently submitted for any degree.

Signed: Hasna Alsaeed Date: 21/10/2025
HASNA JAAFAR MOHAMED AL SAEED (Candidate)

The Thesis defense committee considers the thesis titled:

New Dimension Reduction Methods Based on Auto-Encoder for Efficient Clustering

Submitted by **HASNA JAAFAR MOHAMED AL SAEED** to the College of Information Technology is satisfactory and acceptable for the PhD degree in Computing and Information Sciences.

The Defense Committee:



Dr. Riadh Ksantini
(Supervisor)

Department of Computer Science
University of Bahrain

Prof. Faisal Alkhateeb
(Co-Supervisor)

Department of Computer Science
University of Bahrain



Prof. Nabil Hewahi
(Member)

Department of Computer Science
University of Bahrain



Prof. Boubakeur Boufama
(Member)

School of Computer Science
University of Windsor

ABSTRACT

The rapid growth of time-series data in fields like energy, finance, health-care, and transportation presents significant analytical challenges due to high dimensionality, temporal dependencies, and limited labeled data. Traditional clustering methods often struggle with these complexities, leading to ineffective identification of underlying groupings.

To address these issues, deep clustering methods, particularly those utilizing autoencoders, have become popular. They effectively reduce dimensionality and extract features by compressing high-dimensional data into compact latent representations, capturing essential patterns while minimizing noise. This results in more robust and scalable clustering for complex time-series datasets.

However, deep clustering approaches encounter challenges related to Feature Randomness (FR) and Feature Drift (FD), which can destabilize learned features and shift representations during training, negatively impacting clustering quality. Although recent models have attempted to balance the trade-off between FR and FD, they have not adequately addressed the specific characteristics of time-series datasets.

To effectively balance the trade-off between FR and FD, this thesis proposes two innovative sequential deep embedded clustering models for time series clustering: the Long Short-Term Memory Dynamic Autoencoder (LSTM-DynAE) and the Convolutional Long Short-Term Memory Dynamic Autoencoder (ConvLSTM-DynAE). The LSTM-DynAE model integrates LSTM autoencoders with a dynamic loss function that progressively shifts training focus from reconstruction to clustering, maintaining stable and discriminative representations. The ConvLSTM-DynAE extends this framework by incorporating convolution LSTM (ConvLSTM) autoencoders, enabling simultaneous learning of spatial and temporal features, which is particularly beneficial for load profiling in smart grids.

The proposed models were rigorously validated on both labeled benchmark datasets and real-world smart meter datasets. The LSTM-DynAE achieved an impressive accuracy (Acc) of 0.97 and normalized mutual information (NMI) of 0.95 on the Heartbeat dataset, demonstrating strong clustering performance. The ConvLSTM-DynAE excelled in load profiling, attaining Silhouette Scores(Sil) of 0.980 on the London dataset and 0.996 on the Irish dataset, effectively differentiating consumer energy usage patterns.

These results underscore the effectiveness of deep autoencoder-based dimension reduction and feature extraction in enabling robust, and accurate time-series clustering. By successfully addressing the challenges of FR and FD through dynamic loss functions, the research advances the state-of-the-art in temporal clustering and provides practical tools for smart grid analytics and energy management.

Contents

APPROVAL	A
DECLARATION	B
ABSTRACT	C
TABLE OF CONTENTS	E
LIST OF FIGURES	I
LIST OF TABLES	K
ACKNOWLEDGMENTS	L
DEDICATION	M
PUBLICATIONS	N
LIST OF ABBREVIATIONS	O
1 Introduction	1
1.1 Overview	1
1.2 Background and Motivation	2
1.3 Problem Statement	5
1.4 Research Questions	6
1.5 Aim and Objectives	6
1.6 Contributions	7
1.7 Significance of the Study	8
1.8 Terminology	9
1.9 Thesis Outline	9
1.10 Summary	11

2	Literature Review	12
2.1	Overview	12
2.1.1	Dimension Reduction Methods for Time Series Clustering	13
2.2	Time Series Clustering	18
2.2.1	Challenges of Time Series Clustering	18
2.2.2	Traditional Methods of Time Series Clustering	19
2.2.3	Deep Learning Methods for Time Series Clustering	20
2.3	Auto-Encoder Based Deep Embedded Clustering (DEC) Approaches	23
2.3.1	Separate Embedded Learning and Clustering	24
2.3.2	Integrated Embedded Learning and Clustering	25
2.4	Feature Randomness and Feature Drift in Deep Clustering	27
2.5	Application Example: Load Profiling	30
2.6	Bridging the Gaps	33
2.7	Summary	34
3	Methodology	35
3.1	Overview	35
3.2	Preliminaries	35
3.2.1	Convolutional Neural Network (CNN)	36
3.2.2	Long-Short Term Memory (LSTM)	36
3.2.3	Convolutional Long-Short Term Memory (ConvLSTM)	37
3.3	Proposed Long Short Term Memory- Dynamic Autoencoder (LSTM-DynAE) Model	38
3.3.1	LSTM-AE Architecture	40
3.3.2	Phase-I: Pre-training Phase	41
3.3.3	Phase-II: Clustering Phase	44
3.3.3.1	The Initialization of Clustering Centers	44

3.3.3.2	Dynamic Clustering Loss Function	44
3.3.3.3	Optimization	47
3.3.4	Evaluation Metrics	48
3.4	Proposed ConvLSTM-DynAE Model for Load Profiling . .	51
3.4.1	ConvLSTM-AE Architecture	52
3.4.2	Phase-I: Pretraining Phase	52
3.4.3	Phase-II: Clustering Phase	52
3.4.4	Load Profiling Framework	55
3.4.4.1	Step-1:Data Preprocessing	56
3.4.4.2	Step-2:Clustering	58
3.4.4.3	Step-3:Generating Load Profiles	59
3.5	Summary	62
4	Results and Discussions	64
4.1	Overview	64
4.2	Implementation Settings	64
4.2.1	Implementation Settings of LSTM-DynAE	65
4.2.2	Implementation Settings of ConvLSTM-DynAE . .	65
4.2.2.1	Data Preprocessing	66
4.2.2.2	Clustering	66
4.2.2.3	Evaluation and Generating Load Profiles .	67
4.3	Results of LSTM-DynAE Model	69
4.3.1	Baselines and Datasets	69
4.3.1.1	Baselines	69
4.3.1.2	Datasets	69
4.3.2	Results	70
4.3.2.1	Comparing Results with the State-of-the- Art Methods	70
4.3.2.2	Effect of Dynamic Loss Function	73
4.3.2.3	Feature-Randomness and Feature-Drift . .	74

4.4	Results of ConvLSTM-DynAE Model	76
4.4.1	Baselines and Datasets	77
4.4.1.1	Baselines	77
4.4.1.2	Datasets	77
4.4.2	Results	78
4.5	Discussion	83
4.6	Threats to Validity	89
4.7	Summary	90
5	Conclusion and Recommendations	92
5.1	Overview	92
5.1.1	Conclusion	92
5.2	Recommendations and Future Work	93
	References	95
	Abstract (in Arabic)	

List of Figures

3.1	The Structure of a ConvLSTM Cell	38
3.2	The Architecture of the Proposed LSTM-DynAE Model . .	40
3.3	The Pretraining Phase of LSTM-DynAE Model	43
3.4	The Clustering Phase of LSTM-DynAE Model	44
3.5	The Flowchart of the LSTM-DynAE Model	50
3.6	The Architecture of the proposed ConvLSTM-DynAE Model	51
3.7	Architecture of Convolutional LSTM Autoencoder	53
3.8	The Pretraining Phase of ConvLSTM-DynAE Model	54
3.9	The Clustering Phase of ConvLSTM-DynAE Model	54
3.10	The Flowchart of the ConvLSTM-DynAE Model	56
3.11	The Flowchart of the Load Profiling Framework	62
4.1	Load Profiles of Irish Dataset obtained by ConvLSTM-DynAE	68
4.2	Load Profiles of London Dataset obtained by ConvLSTM-DynAE	68
4.3	Clustering Loss of Dynamic Loss Function During Training on Heartbeat Dataset	74
4.4	Acc and NMI of LSTM-DynAE During the Training of Heartbeat Dataset	74
4.5	Comparison of ΔFR between DTC and LSTM-DynAE During the Training of Heartbeat Dataset	75

4.6	Comparison of ΔFD between DTC and LSTM-DynAE During the Training of Heartbeat Dataset	76
4.7	Elbow methods on Silhouette Score (Sil) for ConvLSTM-DynAE clustering using London dataset	79
4.8	Elbow methods on Silhouette Score (Sil) for ConvLSTM-DynAE clustering using Irish dataset	79
4.9	Sil Score per Iteration for London Dataset	81
4.10	Comparison of ConvLSTM-DynAE Results with Previous Works on Irish Dataset	83
4.11	Comparison of ConvLSTM-DynAE Results with Previous Works on London Dataset	84

List of Tables

2.1	Summary of Deep Embedded Clustering Models	27
2.2	Summary of Recent Works in Load Profiling	32
4.1	Characteristics of the Selected Time Series Datasets	70
4.2	Comparison of the Clustering Performance of LSTM-DynAE with State-of-the-art Clustering Methods in terms of <i>ACC</i> AND <i>NMI</i> Metrics - (Best results in bold, second best emphasized)	72
4.3	Execution Time of LSTM-DynAE and Baseline Models on Benchmark Datasets (in ms)	73
4.4	Silhouette Index (<i>Sil</i>) results for conventional vs deep clustering methods for the London Dataset	80
4.5	Silhouette Index (<i>Sil</i>) results for conventional vs deep clustering methods for the Irish Dataset	80
4.6	Silhouette Score (<i>Sil</i>) for CKSC, DynAE, AEC, and the proposed method (ConvLSTM-DynAE) on the Irish Dataset	82
4.7	Silhouette Score (<i>Sil</i>), DynAE, AEC, and the proposed method (ConvLSTM-DynAE) on the London Dataset	83

ACKNOWLEDGMENTS

Although this journey has been lengthy and fraught with challenges, the satisfaction of achieving my goals brings me immense comfort and joy. Over the nearly five years I have spent on this path, I have faced numerous difficulties and obstacles, yet my determination to succeed has allowed me to overcome these barriers and ultimately enjoy the rewards of my efforts. I would like to sincerely thank my esteemed supervisor, Prof. Riadh Ksantini, and my co-supervisor, Prof. Faisal Al-Khateeb. Your unwavering support has been crucial throughout this arduous journey, providing me with essential guidance and encouragement during my research. With the grace of God Almighty, you have been my strongest advocate, helping me navigate the challenges I encountered while completing this dissertation.

I also want to express my heartfelt appreciation to Dr. Mustafa Hammad and my colleagues at the University of Bahrain for their collaboration and support. Your insights and discussions have greatly enriched my academic experience and made this journey more enjoyable.

Lastly, I wish to thank my family for their endless love, encouragement, and faith in my abilities. Your support has been a constant source of strength, and your patience has served as a beacon of hope throughout this challenging time. I am deeply grateful for your sacrifices and understanding of my circumstances during this demanding period.

DEDICATION

I express my gratitude to God Almighty, the Generous and Merciful, for granting me the strength to complete this endeavor.

I dedicate this thesis to my cherished family, especially my parents, who brought me into this world and taught me to transform life's challenges into opportunities that empower and broaden my perspective. To my beloved husband, Mohammed, my son and hero, Ali, and my two wonderful daughters, Kawthar and Fatima.

This message is also dedicated to you, dear reader. I hope you discover insights within it that illuminate your journey and pave the way for new horizons in the realm of knowledge.

PUBLICATIONS

JOURNAL PAPERS

1. (Submitted to Multimedia Tools and Applications Journal- Status: Under Review): Sequential Deep Embedded Clustering with Dynamic Autoencoder
2. (Submitted to Applied Soft Computing Journal- Status: With Editor): Spatiotemporal Deep Embedded Clustering for Robust Smart Meter Load Profiling

Acronyms

ACC: Accuracy

NMI: Normalized Mutual Information

FR: Feature Randomness

FD: Feature Drift

LSTM: Long Short-Term Memory

CNN: Convolutional Neural Network

ConvLSTM: Convolutional Long Short-Term Memory

Sil: Silhouette Score

DEC: Deep Embedded Clustering

DTC: Deep Temporal Clustering

IDEC: Improved Deep Embedded Clustering

ADEC: Adversarial Deep Embedded Clustering

TMRC: Temporal Multi-Feature Representation Clustering

HEC: Household Energy Consumption

GMM: Gaussian Mixture Model

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

PCA: Principal Component Analysis

UMAP: Uniform Manifold Approximation and Projection

Chapter 1

Introduction

1.1 Overview

The exponential increase in data generation, particularly in the form of time-series data, has heightened the demand for intelligent and scalable analytical frameworks capable of uncovering hidden patterns and structures (X. Wang, Smith, & Hyndman, 2006). Time series data are ubiquitous in various domains, including energy systems, finance, healthcare and transportation, where it plays a crucial role in monitoring, forecasting, and decision-making processes (Långkvist, Karlsson, & Loutfi, 2014). However, the unique characteristics of time-series data—such as high dimensionality, temporal dependencies, and the scarcity of labeled information—pose significant challenges for traditional data mining and machine learning techniques (Jain, 2010). Clustering, a fundamental unsupervised learning task, provides a powerful approach for identifying latent groupings within such data. Yet, existing clustering methods often struggle when applied to sequential datasets (Xie, Girshick, & Farhadi, 2016).

This chapter lays the groundwork for developing a novel deep clustering framework that addresses these limitations. It introduces the research problem, formulates guiding questions, outlines specific objectives, and summarizes the contributions and significance of the proposed work. Additionally,

it provides relevant terminology and a structured overview of how the study advances the field of time-series clustering through the integration of dynamic learning mechanisms and sequential neural architectures.

1.2 Background and Motivation

In today's data-driven world, time series data have emerged as one of the most critical forms of data in various domains, including finance, healthcare, energy systems, transportation and environmental monitoring. Time-series data consists of sequences of observations collected at successive, evenly spaced points in time. Their temporal ordering and dependency structures distinguish it from other types of data, introducing unique analytical opportunities and challenges. As digital transformation accelerates, huge volumes of time series data are continuously generated by sensors, smart devices, and transactional systems. This influx of sequential data underscores the growing need for robust, intelligent, and scalable analytical methods that can extract meaningful insights and patterns over time.

The importance of time-series analysis lies in its ability to capture temporal dynamics and predict future trends based on historical patterns. For example, in healthcare, the analysis of physiological signals such as ECG or glucose levels can enable early diagnosis. In finance, stock price prediction is based on understanding temporal fluctuations. In smart grids, time series analysis allows for better demand forecasting, anomaly detection, and consumer segmentation through load profiling. These applications demand models that can handle high dimensionality, irregular sampling, nonstationarity, and complex dependencies over time - challenges that traditional machine learning algorithms struggle to address effectively.

The key characteristics of time series data include autocorrelation, seasonality, trend components, noise, and variable-length sequences (Hyndman

& Athanasopoulos, 2018). These features make time-series data both rich in information and difficult to model. Moreover, the curse of dimensionality becomes pronounced when dealing with multivariate or high-frequency time-series datasets, leading to increased computational complexity and reduced model performance. Unlike static datasets, time-series data exhibit strong temporal dependencies, where each data point is influenced by preceding values. Capturing such dependencies is critical, yet conventional clustering techniques like K-Means (MacQueen, 1967) or DBSCAN (Ester, Kriegel, Sander, Xu, et al., 1996) typically assume independence between features and instances, making them unsuitable for temporal data.

Another major challenge in time-series analysis is the lack of labeled data, which restricts the use of supervised learning methods. This makes unsupervised learning, particularly clustering, a valuable tool for exploring underlying structures and patterns. However, clustering time-series data presents difficulties, such as defining meaningful similarity measures, coping with noise and missing values, and managing scalability across large datasets. These issues are especially relevant in load profiling, where utilities must group consumer electricity usage patterns without labeled data, often across millions of time series records.

To overcome these limitations, deep learning has become an increasingly popular approach for time series analysis and clustering. Autoencoder architectures (Vincent, Larochelle, Bengio, & Manzagol, 2008) —especially those based on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) - can learn compact latent representations that preserve essential temporal information. These latent representations enable more effective and scalable clustering in high-dimensional sequential data. One particularly promising deep clustering approach is Deep Embedded Clustering (DEC) (Xie et al., 2016), which jointly optimizes feature learning and clustering assignments.

However, DEC-based models and their extensions still face a fundamental challenge: they suffer from Feature Randomness (FR) and Feature Drift (FD). FR refers to the tendency of the encoder to generate random features that lack semantic consistency, while FD describes the shift in learned representations over training epochs, which destabilizes the clustering process.

To address these two problems, a novel model known as DynAE (Dynamic Autoencoder) was proposed by (Mrabah, Khan, Ksantini, & Lachiri, 2019). DynAE introduces a dynamic loss function that gradually shifts the focus of the model from reconstruction (self-supervised) to clustering (pseudo-supervised) as training progresses. This controlled transition helps maintain stable and discriminative feature representations by reducing FR and mitigating FD. Unlike static loss functions, the dynamic nature of DynAE allows the model to adaptively balance the learning objectives at different stages of training, resulting in more robust clustering outcomes.

While DynAE has shown impressive results on static datasets, its original implementation is not tested for modeling temporal dependencies found in sequential data. Consequently, extending DynAE with recurrent structures like LSTM or ConvLSTM becomes necessary to make it suitable for time-series clustering tasks. Moreover, this restricts its direct applicability to time-series problems such as smart meter data profiling, where capturing temporal patterns is critical. Motivated by these insights, this research proposes a novel time series deep clustering framework—LSTM-DynAE and ConvLSTM-DynAE—which integrates the strengths of dynamic loss function with the temporal modeling capabilities of recurrent autoencoders. These models aim to address the full spectrum of challenges associated with time-series clustering, including high dimensionality, temporal dependencies, the absence of labels, and the dual threats of FR and FD. The proposed methods are validated on both labeled benchmark datasets (using *Acc* and *NMI*) and real-world smart meter datasets (using the Silhouette Score (*Sil*)),

demonstrating their practical value and theoretical contributions to the field.

1.3 Problem Statement

Time-series datasets are increasingly central to numerous real-world applications, ranging from financial forecasting and medical diagnostics to energy consumption analysis and environmental monitoring. However, extracting meaningful insights from these datasets poses significant challenges due to their high dimensionality, strong temporal dependencies, and lack of labels. Traditional clustering techniques struggle to capture complex temporal relationships (Shi et al., 2015) and are unable to scale effectively to large, high-dimensional sequential datasets. While deep learning models, particularly autoencoders (Vincent et al., 2008), have demonstrated significant success in learning latent representations for clustering, they are prone to two persistent issues: Feature Randomness (FR) and Feature Drift (FD). These phenomena lead to unstable, non-discriminative representations that degrade clustering quality.

Although DynAE offers a dynamic loss framework to balance reconstruction and clustering and mitigate FR/FD, it was originally developed for static data. Its application to time-series datasets, especially in domains such as load profiling for smart grids, remains limited due to its lack of temporal modeling capabilities. There is a critical need for a sequential deep embedded clustering framework that not only leverages the strengths of dynamic loss but also adapts to the unique nature of temporal data. The sequential datasets used in this study are limited in size due to resource constraints and availability.

1.4 Research Questions

Based on the identified problems, this research aims to answer the following questions:

1. **RQ1:** What are the main challenges associated with clustering high-dimensional time series data?
2. **RQ2:** How will deep learning models, particularly autoencoders, be used for dimensionality reduction in time-series clustering?
3. **RQ3:** How can recurrent architectures like LSTM and ConvLSTM improve the temporal feature extraction process for clustering sequential data?
4. **RQ4:** What are the existing strategies for integrating clustering objectives into representation learning (i.e., Deep Embedded Clustering)?
5. **RQ5:** What is the trade-off between Feature Randomness and Feature Drift in deep clustering, and how has it been addressed in the literature?
6. **RQ6:** How will load profiling in smart grids benefit from clustering and deep learning techniques?
7. **RQ7:** How does the proposed method perform in both labeled sequential data and real-world smart meter load profiling scenarios?

1.5 Aim and Objectives

Aim:

Develop a novel, dynamic, deep clustering model for time-series datasets that balances feature reconstruction and clustering quality while addressing the challenges of Feature Randomness and Feature Drift.

Objectives:

1. **RO1:** To conduct a comprehensive literature review to identify and categorize the key challenges faced in clustering high-dimensional time series data.
2. **RO2:** To develop and evaluate a novel autoencoder architecture specifically designed for effective dimensionality reduction in high-dimensional time series data.
3. **RO3:** To investigate the effectiveness of LSTM and ConvLSTM architectures in improving the extraction of temporal features for improved clustering performance in sequential data.
4. **RO4:** To analyze and compare existing strategies for integrating clustering objectives into representation learning frameworks, identifying strengths and weaknesses.
5. **RO5:** To explore the concepts of Feature Randomness and Feature Drift in deep clustering, and propose methods for mitigating their negative impacts based on literature findings.
6. **RO6:** To assess the impact of clustering and deep learning techniques on the precision and efficiency of load profiling in smart grid applications.
7. **RO7:** To evaluate the performance of the proposed method using sequentially labeled datasets and real-world smart meter data, focusing on accuracy, NMI, and silhouette score.

1.6 Contributions

This research makes the following contributions:

1. Propose a new dimension reduction and clustering model specifically designed for time-series data.
2. Introduce LSTM-DynAE, combining LSTM autoencoders with a dynamic loss mechanism to reduce FR and FD in sequential clustering.
3. Propose ConvLSTM-DynAE for unsupervised load profiling of smart meter data, effectively capturing both spatial and temporal dynamics.
4. Provides a comprehensive empirical evaluation on benchmark sequential datasets and real-world smart meter datasets.
5. Offer a dynamic framework for balancing feature learning and clustering, improving interpretability and clustering robustness.

1.7 Significance of the Study

This research contributes to the field of unsupervised learning for time-series data by providing a scalable and adaptive framework that unifies temporal feature extraction and clustering. The integration of a dynamic loss function within a sequential model addresses key shortcomings in existing deep clustering methods, such as overfitting and instability due to FR and FD. Moreover, the application of ConvLSTM-DynAE to real-world load profiling tasks showcases the practical relevance of the model in the context of smart grid analytics (X. Wang et al., 2006), energy demand forecasting, and consumer segmentation. The framework can be extended to other domains involving sequential data, such as medical diagnostics (Långkvist et al., 2014), natural language processing, and anomaly detection.

1.8 Terminology

- **Time-Series Data:** Data indexed in time order, commonly seen in sensors, stock prices, and smart meters.
- **Autoencoder:** A neural network architecture used for unsupervised representation learning.
- **Long Short-Term Memory (LSTM):** A type of recurrent neural network capable of learning long-term dependencies.
- **Dynamic Autoencoder (DynAE):** A deep clustering model that balances reconstruction and clustering losses using a dynamic cost function.
- **Feature Randomness (FR):** The generation of unstable or meaningless features by the encoder.
- **Feature Drift (FD):** The shifting of latent representations over training, leading to degraded clustering.
- **Load Profiling:** The analysis and categorization of electricity usage patterns from smart meter data.
- **Silhouette Score (Sil):** A metric used to measure the consistency within clusters of data.

1.9 Thesis Outline

This thesis is structured into five main chapters, each addressing critical aspects of the research on deep clustering methods for time-series data. Chapter 1 serves as the introduction, providing an overview of the exponential growth of time-series data and the associated challenges, including high

dimensionality, temporal dependencies, and the scarcity of labeled information. It also outlines the research questions, aims, objectives, and contributions of the study, setting the stage for the subsequent chapters.

Chapter 2 presents a comprehensive literature review, focusing on time-series clustering and the unique challenges posed by high dimensionality and temporal dependencies. It discusses traditional clustering methods, their limitations, and the importance of dimensionality reduction techniques. The chapter highlights advancements in deep learning models for time-series clustering, particularly those utilizing autoencoders, and emphasizes the trade-off between feature randomness and feature drift in deep clustering.

Chapter 3 introduces foundational concepts relevant to the proposed models, including Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and the hybrid architecture of Convolutional LSTM (ConvLSTM). It also details the methodology employed in the study, focusing on the development of the LSTM-DynAE and ConvLSTM-DynAE models. It describes the architecture of the LSTM autoencoder, the pretraining phase, and the clustering phase, including the dynamic loss function that balances the trade-off between feature randomness and feature drift. The chapter also outlines the load profiling framework and the steps involved in data preprocessing, clustering, and generating load profiles.

Chapter 4 provides the results and discussions, showcasing the experimental outcomes of the proposed models on benchmark datasets and real-world smart meter data. It includes a comparison with state-of-the-art methods, an analysis of the effects of the dynamic loss function, and a discussion on feature randomness and feature drift.

Finally, Chapter 5 concludes the thesis by summarizing the key findings, discussing the implications of the research, and providing recommendations for future work in the field of time-series clustering and analysis.

1.10 Summary

This chapter introduced the background, motivation, and research context of clustering in high-dimensional time-series datasets. It outlined the key challenges posed by temporal dependencies, high dimensionality, and unsupervised learning scenarios. The limitations of existing models were discussed, with a focus on FR and FD, leading to the motivation for integrating DynAE with LSTM-based, and ConvLSTM-based architectures. The proposed solution aims to address these challenges through a dynamic, end-to-end deep clustering framework applicable to both labeled sequential data and real-world applications such as smart meter load profiling.

Chapter 2

Literature Review

2.1 Overview

This chapter provides a comprehensive review of the literature highlighting the importance of dimensionality reduction as a preprocessing step, examining both linear and nonlinear techniques, including the emergence of neural network-based approaches like autoencoders. It provides also a detailed review on time series clustering, focusing on the unique challenges posed by high dimensionality and temporal dependencies. It discusses traditional clustering methods and their limitations, such as the curse of dimensionality, noise management, and varying sequence lengths. Furthermore, the chapter explores advancements in deep learning models for time series clustering, detailing the evolution from separate embedded learning and clustering methods to integrated frameworks that optimize both objectives.

A significant focus is placed on the trade-off between feature randomness and feature drift in deep clustering, emphasizing the role of dynamic

loss balancing as a solution to enhance representation stability. The chapter also addresses the application of these methodologies in load profiling within smart grids, underscoring the need for specialized architectures that can effectively capture spatial and temporal interdependencies in energy consumption data. Overall, this chapter positions the study within the context of existing literature, identifying gaps and opportunities for further research in the field of time series clustering.

2.1.1 Dimension Reduction Methods for Time Series Clustering

Dimension reduction is a critical preprocessing step in time-series clustering due to the inherent complexity of temporal data, which often contains noise, high dimensionality, and nonlinear dependencies. Traditional linear methods like Principal Component Analysis (PCA) and Dynamic PCA (Ku, Storer, & Georgakis, 1995) were initially favored for their computational efficiency. However, their reliance on linear projections limits their ability to preserve temporal relationships in nonlinear systems. For instance, (C. Wang, 2017) demonstrated that PCA-based approaches distort temporal dynamics in applications like sensor networks, leading to suboptimal clustering outcomes. This limitation spurred interest in nonlinear methods, such as manifold learning algorithms like Isomap and Locally Linear Embedding (LLE) (Tenenbaum, Silva, & Langford, 2000), which better capture nonlinear structures by preserving local geometries. Despite their theoretical promise, these methods face practical challenges, including sensitivity to noise and an inability to handle streaming data due to their batch-processing nature. (Bengio et al., 2003) highlighted these issues in their study, where Isomap and LLE underperformed due to their reliance on static, noise-free assumptions.

Classical state estimation techniques, such as the Kalman Filter (KF), offer a model-driven approach to noise filtering and recursive latent state estimation, thereby achieving dimension reduction by tracking a low-dimensional state vector. When applying the Kalman filter to systems described by nonlinear state-transition or measurement models, the original filter becomes suboptimal. The two most prominent extensions for handling such nonlinearities are the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). The EKF linearizes the nonlinear functions by employing a first-order Taylor series expansion around the current state estimate, requiring the calculation of Jacobian matrices and potentially leading to divergence or large estimation errors when nonlinearities are severe. Conversely, the UKF utilizes the Unscented Transform (UT), a derivative-free approach that deterministically selects a set of sigma points to capture the true mean and covariance of the state distribution. These sigma points are propagated directly through the actual nonlinear functions. This methodology allows the UKF to achieve significantly higher estimation accuracy—up to the third order of the Taylor series expansion—making it a more robust and preferred choice for highly nonlinear problems in state estimation and data filtering (Julier & Uhlmann, 2004; Wan & Van Der Merwe, 2000).

Despite the strong theoretical foundation of the UKF for noise reduction, this thesis focuses on Auto-Encoders for dimensionality reduction due to two key reasons: 1) The UKF is a model-driven approach, requiring a priori knowledge of the system’s explicit state-space equations (e.g., transition and measurement functions), which are often unknown or prohibitively complex to derive for large, heterogeneous smart-grid and finance datasets. 2) The primary goal of this work is not only dimension reduction but the optimization of the latent space for clustering. Auto-Encoders, particularly those used in Deep Embedded Clustering (DEC) and Dynamic Auto-Encoders

(DynAE), integrate the feature extraction (dimension reduction) and clustering objectives into a single, unified deep learning framework, which a classical filtering technique cannot achieve.

The advent of neural network-based approaches revolutionized dimension reduction for time-series analysis. Among the various methods employed for dimensionality reduction, autoencoders have emerged as a powerful tool due to their ability to capture complex patterns and temporal dependencies inherent in time series data.

Autoencoders (AEs), introduced by (Vincent et al., 2008), emerged as a powerful alternative by learning compressed, low-dimensional representations through reconstruction loss minimization. Autoencoders are a class of neural networks designed to learn efficient representations of data through unsupervised learning. They consist of two main components: an encoder that compresses the input data into a lower-dimensional representation and a decoder that reconstructs the original data from this representation. In the context of time series analysis, LSTM autoencoders are particularly effective, as they can model sequential dependencies. For instance, the time-aware LSTM autoencoder proposed by (Baytas et al., 2017) incorporates mechanisms to handle irregular time intervals, making it suitable for applications in healthcare, such as patient subtyping.

In addition to LSTM autoencoders, Convolutional Neural Network (CNN) autoencoders leverage convolutional layers to extract local features from time series data. Studies have shown that one-dimensional convolutional designs, as demonstrated by (Chen et al., 2024) and (Wachowiak, Moggridge, & Wachowiak-Smolíková, 2019), can effectively capture time-frequency features, leading to improved clustering outcomes. Furthermore, hybrid models that combine CNNs and RNNs, such as the hybrid CNN-GRU autoencoder developed by (Goncalves, Giurgiu, & Schumann, 2020), have demonstrated significant performance improvements. These hybrid

models can jointly extract spatial and temporal features, enhancing the overall representation of the time series data.

Another notable architecture is the Convolutional LSTM (ConvLSTM) autoencoder, which integrates convolutional layers with LSTM units. This architecture is particularly well-suited for spatiotemporal data, as it can capture both spatial and temporal dependencies simultaneously. ConvLSTM autoencoders have been effectively applied in various domains, including video analysis and climate data, where the data exhibits both spatial and temporal characteristics. By utilizing convolutional operations within the LSTM framework, ConvLSTM autoencoders can learn more robust representations of time series data, making them a valuable addition to the suite of dimensionality reduction methods (Shi et al., 2015).

Several techniques are employed within the framework of autoencoders to achieve effective dimensionality reduction. One key technique is feature extraction, where autoencoders learn to extract salient features from the input data, allowing for a more compact representation. This is particularly important in time series analysis, where capturing temporal dependencies is crucial. Additionally, integrating clustering objectives into the training process of autoencoders can enhance the quality of the learned representations. For example, (Zhong, Huang, & Wang, 2023) utilized a multi-layer dilated RNN autoencoder with contrastive learning, achieving notable clustering performance metrics, including a normalized mutual information (NMI) of 0.50 and a Rand index (RI) of 0.77. Moreover, some studies have explored joint optimization of reconstruction and clustering losses, which allows for better cluster separability. This approach is evident in sequence-to-sequence autoencoders, as shown by (Ma, Li, Zhuang, Wang, & Zeng, 2020), where the model learns to reconstruct the input while simultaneously optimizing for clustering.

Despite the promise of deep learning-based dimensionality reduction

methods for time series analysis, several challenges remain. Architectural complexity can lead to difficulties in training and scalability, particularly with large datasets. Additionally, understanding the learned representations and their implications for time series analysis is crucial, especially in domains such as healthcare and finance. Not all models explicitly address temporal mechanisms, which can limit their effectiveness in highly dynamic or irregular time series data. Future research should focus on addressing these challenges, exploring novel architectures, and enhancing the interpretability of deep learning models in the context of time series dimensionality reduction. While autoencoders require more computational resources and tuning than traditional techniques, their flexibility in modeling temporal dependencies, robustness to noise, and compatibility with modern deep learning frameworks solidify their dominance. Challenges such as interpretability and computational cost remain, but these are often outweighed by their performance advantages in complex, real-world applications.

In this thesis, we utilize Long Short-Term Memory Autoencoders (LSTM-AE) and Convolutional LSTM Autoencoders (ConvLSTM-AE) for dimension reduction and feature extraction. This approach is particularly effective as it combines the strengths of both architectures, leveraging the capability of LSTM to model sequential dependencies and the ability of ConvLSTM to capture spatiotemporal features. The integration of these models is facilitated through a deep embedded clustering framework based on autoencoders, which enhances the quality of the learned representations.

2.2 Time Series Clustering

2.2.1 Challenges of Time Series Clustering

Clustering high-dimensional time series data presents a unique set of challenges that significantly hinder the effectiveness of traditional clustering methods such as k-means and dynamic time warping (DTW). One of the primary challenges is the curse of dimensionality, which refers to the phenomenon where the feature space becomes increasingly sparse as the number of dimensions increases. This sparsity can lead to difficulties in identifying meaningful patterns and relationships within the data, as the distance metrics used in clustering become less reliable. High-dimensional settings often exacerbate the impact of irrelevant features and noise, necessitating preprocessing techniques like equal-length transformation to mitigate these issues (Hong, Gu, & Whitehouse, 2017; Tang, Xie, Yang, Niu, & Zhang, 2019). It is important to note that this thesis specifically focuses on high-dimensional time series datasets with fixed lengths, which allows for a more controlled analysis of these challenges.

Another critical challenge is the management of temporal dependencies inherent in time series data. Unlike static datasets, time series data is characterized by temporal relationships that must be explicitly modeled to capture the dynamics of the data accurately. Traditional methods often fail to account for these dependencies, leading to suboptimal clustering results. Recent studies have proposed various approaches, such as cross-predictability and deep learning architectures, to effectively model these temporal dynamics, thereby enhancing the clustering process (Huang, Shah, & Yoo, 2022; Ma, Zheng, Li, & Cottrell, 2019; Trosten, Strauman, Kampffmeyer, & Jenssen, 2019).

Varying sequence lengths also pose a significant challenge in clustering high-dimensional time series data. Time series can vary in length due to

different sampling rates or missing data points, complicating the clustering process. Techniques such as dynamic time warping and shapelet extraction have been developed to address this issue, allowing for the alignment and comparison of sequences of different lengths (Vera-Vera & Roldán-Nofuentes, 2024; Zakaria, Mueen, & Keogh, 2012). However, since our focus is on fixed-length datasets, this particular challenge is less relevant to our analysis, allowing us to concentrate on other aspects of clustering.

Lastly, the presence of noise in time series data can severely impact the clustering results. Noise can arise from various sources, including sensor inaccuracies and environmental factors, leading to distorted data representations. To combat this, researchers have explored robust similarity metrics, such as the L1 norm, and hybrid models that combine statistical and deep learning approaches to enhance noise resilience (Ding et al., 2015; Yin & Yang, 2005). By addressing these challenges, modern clustering methods can achieve improved performance and robustness in high-dimensional, noisy, and irregular time series settings, ultimately leading to more meaningful insights and applications in various domains.

2.2.2 Traditional Methods of Time Series Clustering

Traditional approaches for time series clustering relied heavily on distance-based metrics and manual feature engineering. For instance, DTW introduced by (Berndt & Clifford, 1994), became a cornerstone for aligning time series of variable lengths, enabling clustering via algorithms like K-means. However, the quadratic computational complexity of DTW limited its scalability for large datasets (Dau et al., 2019). Hierarchical clustering methods, such as agglomerative algorithms, provided interpretable dendrogram structures but struggled with high-dimensional data due to their reliance on pairwise distance calculations (Keogh & Kasetty, 2002). Model-based techniques, including Hidden Markov Models (HMMs), offered probabilis-

tic frameworks for sequential data but required stringent assumptions about state transitions and manual feature selection (Aghabozorgi, Shirkhorshidi, & Wah, 2015). Symbolic approaches like Symbolic Aggregate Approximation (SAX) (Lin, Keogh, Lonardi, & Chiu, 2003) improved efficiency by discretizing time series into symbolic representations but sacrificed temporal granularity, reducing their utility in fine-grained clustering tasks.

To address scalability challenges, feature-based methods emerged, focusing on extracting hand-crafted statistical features. (X. Wang et al., 2006) demonstrated the use of simple statistical features (e.g., mean, variance) with K-means clustering. These methods reduced computational costs, but remained constrained by their dependence on domain expertise to engineer relevant features. Hybrid frameworks, such as integrating DTW with density-based clustering (DBSCAN) (Liao, 2005), sought to balance accuracy and efficiency but were hampered by manual parameter tuning and sensitivity to noise.

The computational cost of these methods and their poor performance with high-dimensional datasets make them inefficient for clustering large time series datasets. Therefore, dimension reduction is needed as a prior step before performing these methods for time series clustering.

2.2.3 Deep Learning Methods for Time Series Clustering

The evolution of temporal clustering has been profoundly shaped by the limitations of traditional methods, which struggle with high dimensionality, nonlinear dependencies, and noise in sequential data. Conventional approaches like DTW and K-means rely on handcrafted features or rigid distance metrics, often failing to capture the intricate temporal dynamics of real-world datasets. This shortcoming has spurred the adoption of deep learning, particularly autoencoder-based architectures, which automate feature extraction and model complex temporal patterns. In the following lines,

we discuss key advancements in deep temporal clustering models, their methodologies, and their contributions to the field.

One of the earliest efforts to adapt deep embedded clustering to sequential data is Deep Temporal Clustering (DTC), introduced by (Madiraju, 2018). DTC employs a Long Short-Term Memory (LSTM) autoencoder to map time series into latent representations while jointly optimizing two objectives: temporal reconstruction loss, which preserves sequential fidelity, and clustering loss, which refines cluster assignments. By training these objectives end-to-end, DTC captures long-term dependencies in data such as ECG signals, achieving 89% clustering accuracy on medical datasets. This marked a significant improvement over DTW-based methods, which often failed to model nonlinear temporal relationships. However, the reliance of DTC on LSTMs introduced computational overhead for long sequences, a challenge later addressed by attention-based models.

Building on DTC, Deep Temporal Clustering Representation (DTCR) (Franceschi, Dieuleveut, & Jaggi, 2019) integrates a bidirectional Recurrent Neural Network (RNN) within a sequence-to-sequence (seq2seq) architecture. DTCR combines data reconstruction via the seq2seq framework with K-means loss to generate cluster-specific representations. Unlike DTC, which focuses solely on temporal reconstruction, DTCR explicitly aligns latent embeddings with cluster centroids during training. This dual optimization enhances interpretability, particularly in industrial applications like sensor fault detection, where cluster-specific patterns, such as temperature fluctuations, can be mapped to distinct operational states (Franceschi et al., 2019). However, the dependency of DTCR on K-means limits its flexibility in handling overlapping clusters, a gap later addressed by contrastive learning frameworks.

The Interpretable Multi-Path Time Series Clustering (IT-TSC) model (Xu, Huang, & Yoo, 2021a) addresses the challenge of clustering multi-

variate time series (MTS) with overlapping patterns. IT-TSC employs a multi-path neural network that learns variable association graphs through autoregression. Each cluster corresponds to a unique path in the network, and cluster assignments are determined by the path with the smallest regression error. This architecture not only improves clustering accuracy but also provides interpretable insights into how variables interact within each cluster. For instance, in energy consumption data, IT-TSC can reveal that Cluster A is characterized by strong correlations between temperature and power usage, while Cluster B reflects time-lagged relationships (Xu et al., 2021a). This interpretability makes IT-TSC particularly valuable in domains like industrial process monitoring, where understanding cluster distinctions is as critical as accuracy.

The Self-supervised Time Series Clustering Network (STCN) (Ma et al., 2020) introduces a novel framework that unifies feature extraction and clustering through self-supervision. STCN uses an RNN to capture temporal dynamics and preserve local structures via one-step-ahead prediction. The hidden states of RNN are then fed into a self-supervised clustering module that iteratively refines cluster assignments. By leveraging prediction errors as a form of reconstruction loss, STCN achieves 85% purity on motion sensor datasets, outperforming methods like DTCR that rely solely on K-means (Ma et al., 2020). The strength of STCN lies in its ability to handle irregularly sampled data, a common issue in IoT and wearable sensor applications.

Deep Temporal Contrastive Clustering (DTCC) (Zhong et al., 2023) represents a paradigm shift by integrating contrastive learning with autoencoder-based clustering. DTCC employs dual temporal autoencoders to generate two augmented views of the input data, such as time-warped and noise-injected versions. It then applies sample-wise and cluster-wise contrastive losses to maximize agreement between similar instances while separating dissimilar ones. This approach enhances feature distinctiveness, en-

abling DTCC to achieve 25% higher normalized mutual information (NMI) than DTW-Kmeans on financial datasets. Additionally, the unified objective function of DTCC combining reconstruction, clustering, and contrastive losses—eliminates the need for pretraining, making it computationally efficient.

The Temporal Multi-Feature Representation Clustering (TMRC) model (Lee, Kim, & Sim, 2024) tackles the complexity of real-world energy consumption data, which often contains overlapping temporal patterns, such as daily and seasonal trends. TMRC first decomposes time series into intrinsic mode functions using Variational Mode Decomposition (VMD), separating distinct temporal components. These components are then processed by dedicated LSTM autoencoders to extract cluster-specific features. By combining decomposition with deep clustering, TMRC improves anomaly detection accuracy in smart grids by 18% compared to monolithic architectures like DTC (Lee et al., 2024). This hybrid approach bridges the gap between signal processing and deep learning, offering a blueprint for handling multifrequency temporal data.

In conclusion, the shift from traditional to deep learning-based temporal clustering has been driven by autoencoder architectures and their derivatives—DTC, DTCR, IT-TSC, STCN, and DTCC—which address the unique challenges of sequential data through innovations in reconstruction, and interpretability. These models have demonstrated superior performance in many areas such as healthcare, energy, and finance,...etc.

2.3 Auto-Encoder Based Deep Embedded Clustering (DEC) Approaches

Autoencoder-based deep clustering has emerged as a cornerstone of unsupervised learning, addressing the limitations of traditional clustering meth-

ods that rely on handcrafted features or simplistic distance metrics. Current auto-encoder based deep clustering techniques can be classified into two core categories based on their approach to embedded learning and clustering assignments: (1) Separate Embedded Learning and Clustering, and (2) Integrated Embedded Learning and Clustering. These categories reflect distinct methodologies in how models utilize embedded representations and clustering assignments to enhance clustering performance, ultimately influencing their effectiveness in various applications. Following lines present each category and discuss some major methods belong to each of them.

2.3.1 Separate Embedded Learning and Clustering

In the Separate Embedded Learning and Clustering approach, embedded learning occurs prior to clustering, and the newly obtained labels from the clustering stage are not utilized to refine the discriminative features. This methodology emphasizes the initial learning of representations without direct feedback from the clustering process, which can limit the ability of the model to adaptively improve its feature extraction based on clustering outcomes.

One prominent model in this category is Deep Embedded Clustering (DEC). DEC employs a deep neural network architecture that focuses on learning feature representations through iterative optimization. The model is designed to learn a mapping from the input space to a lower-dimensional latent space, where clustering is performed. However, DEC does not leverage the clustering assignments to refine the learned representations, which can lead to suboptimal clustering performance. The iterative optimization process allows DEC to improve its clustering results over time, but the lack of integration between the learning and clustering phases means that the model may miss opportunities to enhance its feature extraction based on the clustering outcomes (Xie et al., 2016).

Another model in this category is Adversarial Deep Embedded Clustering (ADEC). ADEC utilizes an adversarial autoencoder to learn representations without explicitly refining them based on clustering assignments. The model focuses on distribution matching in the latent space through adversarial training, which helps preserve the data structure. While ADEC effectively learns robust representations, it does not utilize the clustering results to improve the feature representations, limiting its adaptability to the underlying data distribution (Zhou & Zhou, 2019). This separation of learning and clustering can hinder the overall performance of the model, particularly in complex datasets where the relationships between data points are crucial for effective clustering.

2.3.2 Integrated Embedded Learning and Clustering

In contrast, the Integrated Embedded Learning and Clustering approach involves using a traditional clustering algorithm to group the embedded data points into distinct clusters. Subsequently, the encoding transformation is trained based on the clustering assignments to create a more clustering-friendly representation. This methodology allows for a more dynamic interaction between the learning and clustering processes, leading to improved clustering performance. One of the key models in this category is Deep Embedded Regularized Clustering (DEPICT). DEPICT employs a multi-layer convolutional autoencoder architecture combined with multinomial logistic regression. This model integrates both clustering and reconstruction objectives through the minimization of Kullback-Leibler (KL) divergence, ensuring that the learned representations maintain fidelity to the original data while effectively clustering it. The pretraining phase involves training the autoencoder to reconstruct the input data, followed by the clustering phase where the model optimizes the clustering assignments based on the learned representations. This dual approach allows DEPICT to achieve superior

clustering performance while being robust to overfitting, as it effectively utilizes the clustering results to refine the learned features (Ghasedi Dizaji, Herandi, Deng, Cai, & Huang, 2017).

Another notable model is Improved Deep Embedded Clustering (IDEC), which emphasizes local structure preservation by jointly optimizing clustering and reconstruction losses. IDEC utilizes an under-complete autoencoder alongside a multilayer perceptron (MLP). The pretraining phase focuses on the ability of the autoencoder to reconstruct the input data, while the clustering phase refines the clustering assignments based on the learned features. This integration of clustering assignments into the learning process ensures that the clusters reflect the underlying data distribution, leading to improved clustering outcomes (Guo, Gao, Liu, & Yin, 2017a).

Joint Unsupervised Learning is another model that exemplifies the integrated approach. This model combines a convolutional neural network (CNN) with agglomerative clustering in a recurrent framework. It employs a weighted triplet loss during the pretraining phase to enhance the feature representations, followed by a clustering phase where the learned features are used for clustering. The integration of both encoder and decoder allows for a robust representation that captures both local and global structures in the data, leading to strong generalization across various datasets (J. Yang, Parikh, & Batra, 2016).

Lastly, Statistical Deep Embedded Clustering (StatDEC) integrates a deep statistics network with a clustering network, utilizing a statistics pooling layer to handle class imbalance and out-of-distribution samples. The pretraining phase focuses on learning robust representations, while the clustering phase integrates these representations with statistical methods to enhance clustering performance. This model addresses challenges related to class imbalance and out-of-distribution data, making it particularly effective in real-world applications (Rezaei, Dorigatti, Ruegamer, & Bischl, 2021).

Table 2.1 shows a summary of the deep embedded clustering models with their categories.

Table 2.1: Summary of Deep Embedded Clustering Models

Study Reference	Model Name	Category	Key Features
(Xie et al., 2016)	Deep Embedded Clustering (DEC)	Separate Embedded Learning and Clustering	Deep neural network, iterative optimization
(Zhou & Zhou, 2019)	Adversarial Deep Embedded Clustering (ADEC)	Separate Embedded Learning and Clustering	Adversarial autoencoder, distribution matching
(Ghasedi Dizaji et al., 2017)	Deep Embedded Regularized Clustering (DEPICT)	Integrated Embedded Learning and Clustering	Multi-layer convolutional autoencoder, KL divergence minimization
(Guo et al., 2017a)	Improved Deep Embedded Clustering (IDEC)	Integrated Embedded Learning and Clustering	Under-complete autoencoder, local structure preservation
(J. Yang et al., 2016)	Joint Unsupervised Learning	Integrated Embedded Learning and Clustering	CNN with agglomerative clustering, weighted triplet loss
(Rezaei et al., 2021)	Statistical Deep Embedded Clustering (StatDEC)	Integrated Embedded Learning and Clustering	Deep statistics network, statistics pooling for class imbalance

2.4 Feature Randomness and Feature Drift in Deep Clustering

In deep clustering, a critical challenge is the trade-off between feature randomness (FR) and feature drift (FD). Feature randomness refers to the introduction of diversity in the learned features, which can lead to noisy and non-representative representations. This often occurs when fixed-weight loss functions are employed, as they may not adapt to the varying complexities of the data, resulting in features that do not accurately capture the underlying structure of the dataset (Mrabah, Bouguessa, & Ksantini, 2020). Conversely, feature drift occurs when the learned representations systematically shift away from optimal clustering due to conflicting training objectives,

particularly when combining clustering with reconstruction losses (Mrabah et al., 2019). This drift can undermine the effectiveness of the clustering process, leading to suboptimal performance.

The interaction between self-supervision and pseudo-supervision further complicates this trade-off. Previous studies have demonstrated that conventional deep clustering methods are significantly influenced by the balance between FR and FD (Zhou & Zhou, 2019; Guo et al., 2017a). The FR issue is particularly associated with the pseudo-supervision task; specifically, using pseudo-labels to learn features aimed at clustering can mislead the trained network. While some pseudo-labels may align with the actual ground-truth labels, others do not correspond to the true categories, leading the network to learn random features that capture irrelevant similarities. For example, the DEC model utilizes pseudo-supervision without incorporating an auxiliary self-supervision task during the fine-tuning phase, resulting in a lack of mechanisms to prevent random projections induced by pseudo-supervision.

To alleviate the impact of random features, several deep clustering models employ self-supervision strategies not only as a pretraining task but also as an auxiliary task during the second phase. Reconstruction has become the standard self-supervision technique for existing deep clustering models. However, the discriminative features learned through the clustering objective can easily drift due to the self-supervision task. The FD issue arises from the intense competition between self-supervision and pseudo-supervision tasks. For instance, the Improved Deep Embedding Clustering model (IDEC) enhances DEC by jointly performing clustering and standard reconstruction in the second phase, yet it still experiences FD. The clustering loss aims to minimize within-cluster variance while maximizing between-cluster variance, whereas the standard reconstruction loss seeks to maintain both within and between-cluster variances.

The first work proposed to examine the balance between FR and FD was Adversarial Deep Embedded Clustering (ADEC) (Mrabah, Bouguessa, & Ksantini, 2020). ADEC pioneers adversarial regularization within autoencoder-based clustering to explicitly balance the representation of FR and FD. The model minimizes KL divergence relative to an auxiliary target distribution while enforcing clustering objectives through adversarially constrained reconstruction. Despite mitigating conflicts between these dual goals, ADEC suffers from three inherent instability limitations. First, the Mode collapse, which restricted generative diversity to subsets of the true data distribution; Second, Non-convergence tendencies; and the third limitation is Memorization effects that impede extrapolation beyond training samples. Dynamic loss balancing has emerged as a promising solution to address this trade-off between FR and FD. By gradually adjusting the emphasis on reconstruction loss during training, dynamic loss schemes can effectively manage the balance between FR and FD. For instance, Deep Clustering with a Dynamic Auto-Encoder (DynAE) progressively adjusts the self-supervision task. The training process of DynAE is divided into two phases. In the pretraining phase, the network engages in a pretext task (i.e., reconstruction with adversarially constrained interpolation). In the subsequent clustering phase, the network is fine-tuned using a dynamic blend of self-supervision and pseudo-supervision. This dynamic learning mechanism involves gradually transitioning the reconstruction loss into a clustering-oriented self-supervision task. Specifically, the model generates decoded centroid images for high-confidence samples (i.e., those with low-entropy clustering assignment scores) and applies standard reconstruction for low-confidence samples (i.e., those with high-entropy clustering assignment scores). As a result, the loss function evolves progressively during the clustering phase to diminish the impact of FD while avoiding excessive FR. Therefore, DynAE has shown promising outcomes by mitigating the

balance between FR and FD.

While dynamic loss balancing has primarily been applied to non-temporal datasets, this study extends its use to temporal datasets. These data pose unique challenges, including capturing dependencies and variations over time. By employing dynamic loss balancing, we aim to reduce the trade-off between feature randomness and feature drift, yielding representations that are both diverse and stable. This is expected to improve clustering performance on temporal data, enabling more accurate and meaningful insights.

2.5 Application Example: Load Profiling

Although this thesis treats load profiling merely as one experimental dataset for validating the proposed ConvLSTM-DynAE model, it remains an illustrative application area for testing clustering and deep-representation methods on large-scale time-series data.

Early studies used traditional clustering algorithms such as K-Means, Hierarchical Clustering, and Gaussian Mixture Models (GMM) for grouping consumers based on daily or seasonal consumption curves (Y. Wang et al., 2015; McLoughlin, Duffy, & Conlon, 2013). (Panapakidis & Christoforidis, 2017) enhanced K-Means initialization to mitigate sensitivity to local minima. (Rajabi et al., 2017, 2020) compared several clustering approaches, including K-Means, Fuzzy C-Means (FCM), Hierarchical, and SOM, across smart-meter datasets, identifying trade-offs in compactness and interpretability. (Yilmaz, Chambers, & Patel, 2019) empirically compared K-Means, K-Medoids, and Hierarchical clustering, linking cluster quality to demand-side-management outcomes. (Zarabie, Lashkarbolooki, Das, Jhala, & Pahwa, 2019) explored Affinity Propagation to automatically determine cluster numbers, while (Al Khafaf, Jalili, & Sokolowski, 2020)

proposed a new clustering-index metric for optimal cluster-count selection.

To address scalability and data heterogeneity, other studies introduced adaptive and ensemble approaches. (Le Ray & Pinson, 2019) proposed an online adaptive K-Means algorithm to handle continuous smart-meter streams. (Sun, Zhou, & Yang, 2020) combined concurrent K-Means, spectral, and ensemble clustering for robustness. (Czétány et al., 2021) tested multiple traditional algorithms (K-Means, FCM, and HAC) on Hungarian smart-meter data, confirming method-specific performance differences.

With the emergence of deep learning, researchers began developing deep-embedded and autoencoder-based clustering techniques for more expressive load-pattern discovery. (Eskandarnia, Al-Ammal, & Ksantini, 2022) applied Deep Embedded Clustering (DEC), Improved DEC (IDEC), and DynAE models to the London and Irish datasets, demonstrating superior Silhouette scores and cluster stability compared to static autoencoders. (W. Yang, Li, Chen, & Hong, 2022) employed LSTM Autoencoders to extract multi-scale temporal features. (Okereke et al., 2023) incorporated time-domain feature extraction prior to K-Means clustering on smart-meter data. (Kumar & Mallipeddi, 2024) proposed a deep clustering framework unifying feature learning and clustering objectives for large residential datasets.

Collectively, these works (summarized in Table 2.2) demonstrate the methodological transition from classical clustering to deep-representation learning frameworks capable of capturing nonlinear temporal dependencies and complex consumer behaviors in electricity-consumption data.

In the context of this thesis, load profiling is not analyzed as an independent research theme but serves as a benchmark application for assessing the clustering capability of the proposed ConvLSTM-DynAE model using large-scale, real-world smart-meter datasets (London and Irish). The detailed preprocessing and clustering steps for these datasets are described in

Chapter 3, Section 3.4.4.

Table 2.2: Summary of Recent Works in Load Profiling

Reference	Dataset & Models	Metrics	Dataset Size	Pros & Cons
(Panapakidis & Christoforidis, 2017)	Sample data; Modified k-means clustering	WCBCR, SI, IAI, IEI	150×1 yr	Pros: real-consumer data; simple k-means $\rightarrow$ easy to replicate. Cons: small sample size, limited variety, only one year.
(Rajabi et al., 2017)	Irish dataset; Survey of clustering methods	Various, based on reviewed studies	5225×1 yrs	Pros: broad survey across methods. Cons: no large new dataset; more review than new empirical work.
(Le Ray & Pinson, 2019)	Synthetic dataset ($\sim 13k$ loads); Adaptive k-means	Sil	First: $\sim 7k \times 1$ month, Second: 13241×1 yr	Pros: control over synthetic data, method testing. Cons: synthetic (not real) data.
(Yilmaz et al., 2019)	Swiss domestic electricity; K-means	CHI, C Index, DBI, etc.	656×1 yr	Pros: real smart-meter data; good dataset size. Cons: limited geographic diversity.
(Zarabie et al., 2019)	Pecan Street; Affinity propagation	Sil, CHI, DBI, DI, etc.	$\sim 1000 \times 1$ yr	Pros: advanced clustering (affinity propagation). Cons: less common method.
(Al Khafaf et al., 2020)	Two real-world datasets; Proposed clustering index	CHI, DBI, Modified DBI, GS, etc.	$\sim 5000 \times 1$ yr	Pros: introduces new clustering index. Cons: dataset details unclear.
(Rajabi et al., 2020)	Irish dataset; K-means, FCM, hierarchical, SOM	Sil	$\sim 5000 \times 2$ yrs	Pros: applies multiple clustering techniques. Cons: dataset size not specified.
(Sun et al., 2020)	Irish dataset; Concurrent k-means, spectral, ensemble clustering	Sil	$\sim 5000 \times 2$ yrs	Pros: ensemble methods increase robustness. Cons: dataset size unspecified.
(Cz�t�ny et al., 2021)	Hungarian smart meter data; K-means, FCM, HAC	DI, Sil, Elbow method	24917×1 yr	Pros: multiple clustering methods; smart-meter data. Cons: size and sampling not clear.
(Eskandarnia et al., 2022)	London and Irish datasets; DEC, IDEC, DynAE	Sil, Load curve analysis	London: $\sim 5600 \times 3$ yrs, Irish: $\sim 5000 \times 2$ yrs	Pros: two geographic datasets. Cons: method complexity; size unspecified.
(W. Yang et al., 2022)	Residential multi-scale electricity use; LSTM autoencoder	MAE, RMSE, MAPPE	35040×1 yr	Pros: uses deep-learning autoencoder. Cons: evaluation metrics more forecasting oriented.
(Okereke et al., 2023)	Smart meter data; K-means with time-domain features	Sil, DBI, Elbow method	London: $\sim 5600 \times 3$ yrs	Pros: large real dataset; strong empirical basis. Cons: only one region (London).
(Kumar & Mallipeddi, 2024)	Residential datasets; Deep clustering framework	Sil	London: $\sim 5600 \times 3$ yrs, Irish: $\sim 5000 \times 2$ yrs	Pros: latest deep-clustering application. Cons: dataset size unspecified.
This Study	London (~ 5600 houses) and Irish (~ 5000 homes); ConvLSTM-AE, DEC, DynAE, LSTM-AE, etc.	Sil, Load curve analysis	London: $\sim 5600 \times 3$ yrs; Ireland: $\sim 5000 \times 2$ yrs	Pros: large multi-dataset, multi-method comparison. Cons: method diversity may dilute focus.

2.6 Bridging the Gaps

The current literature on dynamic loss balancing in temporal datasets reveals several gaps, particularly the need for more comprehensive methodologies that address the unique challenges of temporal dependencies and high dimensionality. Furthermore, there is a scarcity of studies that explore the integration of advanced architectures, such as LSTM-based models, to improve performance in high-dimensional temporal contexts. This study aims to bridge these gaps by extending the application of dynamic loss balancing from primarily non-temporal datasets to temporal datasets, which require effective strategies for capturing temporal dependencies and managing variations over time. To achieve this, we utilize an LSTM-based autoencoder (LSTM-AE), an architecture specifically designed to capture the inherent temporal dependencies in sequential data. The LSTM-AE also adeptly handles high-dimensional datasets through dimensionality reduction and deep embedded clustering. By incorporating dynamic loss balancing within this framework, we seek to alleviate the trade-off between feature randomness and feature drift, ensuring that the learned representations are both diverse and stable. This approach is anticipated to enhance clustering performance on temporal datasets, leading to more accurate and meaningful insights.

Similarly, the load profiling domain presents unique challenges that demand specialized architectural solutions. Traditional encoder-decoder structures often process spatial and temporal features sequentially, which can limit their ability to capture the intricate interdependencies found in energy consumption data. This limitation has prompted the development of a Convolutional LSTM (ConvLSTM) autoencoder, which integrates convolutional operations directly within LSTM cells to simultaneously model spatial and temporal relationships. The integration of ConvLSTM in both the encoder and the decoder marks a significant advancement, allowing the concurrent processing of spatial correlations and temporal dependencies in energy data.

Our study aims to take advantage of the strengths of the LSTM-AE to address similar challenges in temporal data sets. This is particularly important for capturing temporal dependencies and variations. By employing this architecture alongside dynamic loss balancing, we can effectively tackle high-dimensional datasets while ensuring that the learned representations remain diverse and stable. This approach parallels the advances achieved with ConvLSTM in energy load profiling. We expect it to significantly enhance clustering performance on temporal datasets, ultimately yielding more accurate and insightful interpretations of the data.

2.7 Summary

This chapter provided a comprehensive overview of the challenges and methodologies in time series clustering. It particularly focused on the integration of advanced techniques such as deep embedded clustering and dynamic loss balancing. It highlighted the limitations of traditional methods and the need for innovative approaches to effectively manage high-dimensional, noisy, and temporally dependent data. Using architectures such as LSTM-AE and ConvLSTM-AE, this study aims to improve clustering performance in temporal datasets, especially in the context of load profiling within smart grid applications. The findings underscore the importance of addressing the trade-off between feature randomness and feature drift, ultimately contributing to more accurate and meaningful insights from time series data.

Chapter 3

Methodology

3.1 Overview

This chapter introduces some foundational preliminaries, and the two deep embedded clustering models developed in this study. The first section provides some definitions of important concepts needed for the proposed models. The second section describes the LSTM-DynAE model, and organized into three subsections: the architecture of the LSTM autoencoder, the pretraining phase, and the clustering phase. The clustering phase subsection elaborates on the dynamic loss function in detail. The third section presents the ConvLSTM-DynAE model, designed for load profiling. It is divided into subsections covering the ConvLSTM autoencoder architecture and the two main phases of the model. The final subsection outlines the complete steps of the load profiling framework.

3.2 Preliminaries

In this section, we present some foundational concepts to facilitate understanding of the description of our proposed models.

3.2.1 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a type of feedforward deep neural network primarily used for analyzing visual data (Valueva, Nagornov, Lyakhov, Valuev, & Chervyakov, 2020). It has a diverse range of applications, including image and video recognition, image classification, and recommendation systems. Unlike traditional neural networks, a CNN employs convolution operations instead of general matrix multiplication in at least one of its layers, allowing it to significantly reduce the number of parameters without sacrificing model quality. As the network progresses through its layers, it learns more complex and higher-dimensional features. The input data layer accepts an input vector shaped as $n \times h \times w \times c$, where n represents the number of images, h is the height, w is the width, and c denotes the number of channels. This input is then fed into the convolutional layer, which applies convolution to the input and generates a feature map based on a specific set of weights.

3.2.2 Long-Short Term Memory (LSTM)

An LSTM (Long Short-Term Memory) deep neural network is a specialized form of Recurrent Neural Networks (RNNs) that is particularly adept at modeling temporal patterns, such as those found in time series, language, audio, and text. This capability arises from the feedback loops utilized for learning and the additional parameters that facilitate connections between time steps. The key components of an LSTM include memory cells and the input, forget, and output gates. These components enable the LSTM network to maintain connections from previous time steps and layers, allowing each output to be influenced by both the current input and historical inputs. Typically, LSTM architectures consist of multiple layers, each layer containing several LSTM units, and each unit comprising the three gates mentioned above (e.g., input, forget and output gates) (Tayeh, Aburakhia, & Myers, 2022). Equations from 3.1 to 3.6 governing the input, forget, and output gates, as well as the candidate cell state, the cell state, and the LSTM cell output, are defined as follows:

$$\textbf{Input Gate: } g_t = \sigma(W_{hg}h_{t-1} + W_{xg}x_t + b_g) \quad (3.1)$$

$$\textbf{Forget Gate: } f_t = \sigma(W_{hf}h_{t-1} + W_{xf}x_t + b_f) \quad (3.2)$$

$$\textbf{Output Gate: } o_t = \sigma(W_{ho}h_{t-1} + W_{xo}x_t + b_o) \quad (3.3)$$

$$\textbf{Candidate Cell State: } \tilde{C}_t = \tanh(W_{hC}h_{t-1} + W_{xC}x_t + b_C) \quad (3.4)$$

$$\textbf{Cell State: } C_t = f_t \odot C_{t-1} + g_t \odot \tilde{C}_t \quad (3.5)$$

$$\textbf{Hidden State: } h_t = o_t \odot \tanh(C_t) \quad (3.6)$$

In these equations, $\odot$ signifies the Hadamard (element-wise product) function, g_t represents the input gate, f_t denotes the forget gate, o_t indicates the output gate, $\tilde{C}_t$ is the candidate cell state, C_t is the cell state, and h_t is the hidden state and cell output at time step t . The input is represented by the notation x_t . The notation W and b represent the weight matrix and the bias vectors to be optimized, respectively. The activation functions of the sigmoid and tanh are given by $\sigma(x) = \frac{1}{1+e^{-x}}$ and $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The input gate is designed to capture useful information from both the input data and the previous hidden state h_{t-1} . The forget gate f_t is tasked with removing information that is no longer necessary or relevant, while the output gate o_t regulates the flow of information.

3.2.3 Convolutional Long-Short Term Memory (ConvLSTM)

ConvLSTM is a hybrid deep neural network architecture that serves as a variant of LSTM (Shi et al., 2015). The ConvLSTM has convolutional structures in both the input-to-state and state-to-state transitions. It determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. This can easily be achieved by using a convolution operator in the state-to-state and input-to-state transitions by incorporating convolution operations within the LSTM cell to effectively capture the spatial features present in higher-dimensional input data. A representation of a ConvLSTM cell is shown in Figure 3.1.

The key equations of ConvLSTM are shown by the following Equations from 3.7 to

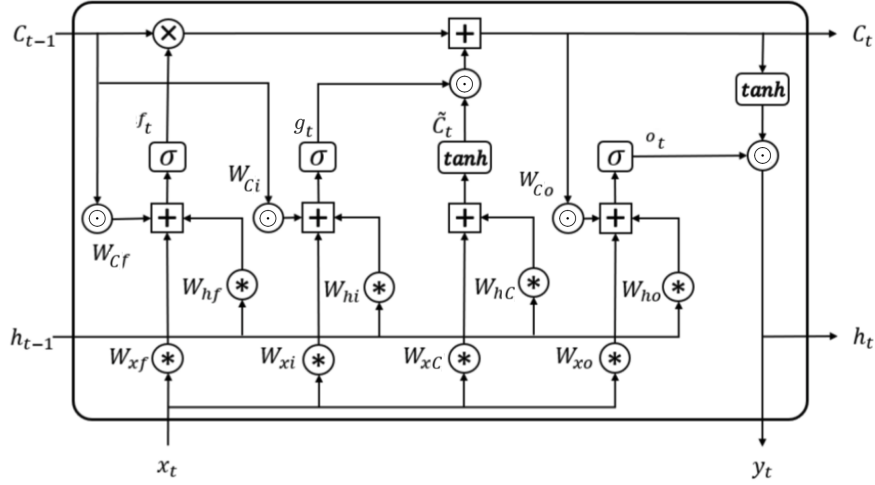


Figure 3.1: The Structure of a ConvLSTM Cell

3.12, where $*$ denotes the convolution operation:

$$\textbf{Input Gate: } g_t = \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + W_{ci} \odot C_{t-1}) \quad (3.7)$$

$$\textbf{Forget Gate: } f_t = \sigma(W_{xf} * x_t + W_{hf} * h_{t-1} + W_{cf} \odot C_{t-1}) \quad (3.8)$$

$$\textbf{Output Gate: } o_t = \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + W_{co} \odot C_t) \quad (3.9)$$

$$\textbf{Candidate Cell State: } \tilde{C}_t = \tanh(W_{xc} * x_t + W_{hc} * h_{t-1}) \quad (3.10)$$

$$\textbf{Cell State: } C_t = f_t \odot C_{t-1} + g_t \odot \tilde{C}_t \quad (3.11)$$

$$\textbf{Hidden State: } h_t = o_t \odot \tanh(C_t) \quad (3.12)$$

3.3 Proposed Long Short Term Memory- Dynamic Autoencoder (LSTM-DynAE) Model

In this section, we introduce our proposed sequential dynamic autoencoder (LSTM-DynAE) model which is shown in Figure 3.2. LSTM-DynAE is a double-stage deep clustering model. In Section 3.3.1, we illustrate the architecture of the LSTM Autoencoder utilized in our model. Section 3.3.2 provides a detailed explanation of the pretraining of our model. In Section 3.3.3, we present an overview of the second phase, referred to as the clustering phase. Section 3.3.4, presents the evaluation metrics used to evaluate the proposed model.

The pretraining stage is crucial as it allows the neural network to learn general-purpose

features that serve as an initialization. We previously indicated that pseudo-labels extracted from random latent representations are unreliable. Therefore, one can posit that random pseudo-labels do not provide a strong supervisory signal. This randomness can lead to excessive feature variations, which misguides the entire learning process. As a solution, we recommend pretraining the neural network on a pretext task to acquire valuable insights before transitioning to pseudo-supervised training. This approach facilitates the learning of discriminative representations and allows for the introduction of effective pseudo-supervision. For the clustering phase, we outline the dynamic loss function integrated with the model which has been proposed by (Mrabah et al., 2019) in DynAE. The dynamic learning mechanism strikes a better balance between feature randomness and feature drift by gradually and smoothly shifting from the reconstruction objective function to a construction-oriented one.

Before delving into the proposed method, essential notations are defined. Let $X = \{x_i \in \mathbb{R}^d\}_{i=1}^N$ represent a time series dataset utilized for the clustering task, where N denotes the number of data vectors in $\mathbb{R}^d$. The objective of this task is to form K clusters denoted as $\{C_k\}_{k=1}^K$.

In the context of Autoencoder (AE)-based clustering, the encoder mapping is described by $f_{\phi_e} : X \rightarrow Z$, while the decoder mapping is defined by $g_{\phi_d} : Z \rightarrow X$. Here, ϕ_e and ϕ_d are the learnable parameters associated with the encoder and decoder mappings, respectively. The latent representation is noted as z_i and the reconstructed representation for a given data vector x_i as $\hat{x}_i$, where $x_i = \{x_i^1, x_i^2, \dots, x_i^T\}$ is a vector of length T . Thus, we have $z_i = f_{\phi_e}(x_i) \in \mathbb{R}^d$ and $\hat{x}_i = g_{\phi_d}(z_i) \in \mathbb{R}^d$, where $\hat{x}_i = \{\hat{x}_i^T, \hat{x}_i^{T-1}, \dots, \hat{x}_i^1\}$ represents the reconstructed vector produced by the decoder, essentially a reverse of the input.

To conduct clustering, prior knowledge of the number of clusters K and the set of latent centroids is necessary. The set of latent centroids is denoted by $\eta = \{\mu_r \in \mathbb{R}^p\}_{r=1}^K$, where μ_r serves as the latent centroids within the latent representation. The clustering assignments are determined by evaluating the similarity between the latent centroids μ_r and the embedded data points z_i . A Student's t-distribution (Hurst, 2010) is employed as a kernel to evaluate similarity between embedded points and centroids, as shown in Equation 3.13. Here, q_{ir} represents the soft clustering assignments for the embedded data points z_i toward the centroids μ_r , α is a coefficient specific to the Student's t-distribution, and Q is a matrix defined as $Q = (q_{ir})$, where $i = 1, \dots, N$ and $r = 1, \dots, K$. The encoder function f_{ϕ_e} is denoted by σ , indicating the output of $\sigma(x_i)$ with the highest clustering assignment as illustrated in Equation 3.14. Therefore, we have

$$q_{ir} = \frac{\left(1 + \frac{\|z_i - \mu_r\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_{r'} \left(1 + \frac{\|z_i - \mu_{r'}\|^2}{\alpha}\right)^{-\frac{\alpha+1}{2}}}. \quad (3.13)$$

$$\sigma(x_i) = \mu_{\arg\max_r q_{ir}} \quad (3.14)$$

LSTM-DynAE is designed to determine ϕ_e that allows the transformation function f to project the data into a clustering-friendly embedded space based on the chosen kernel.

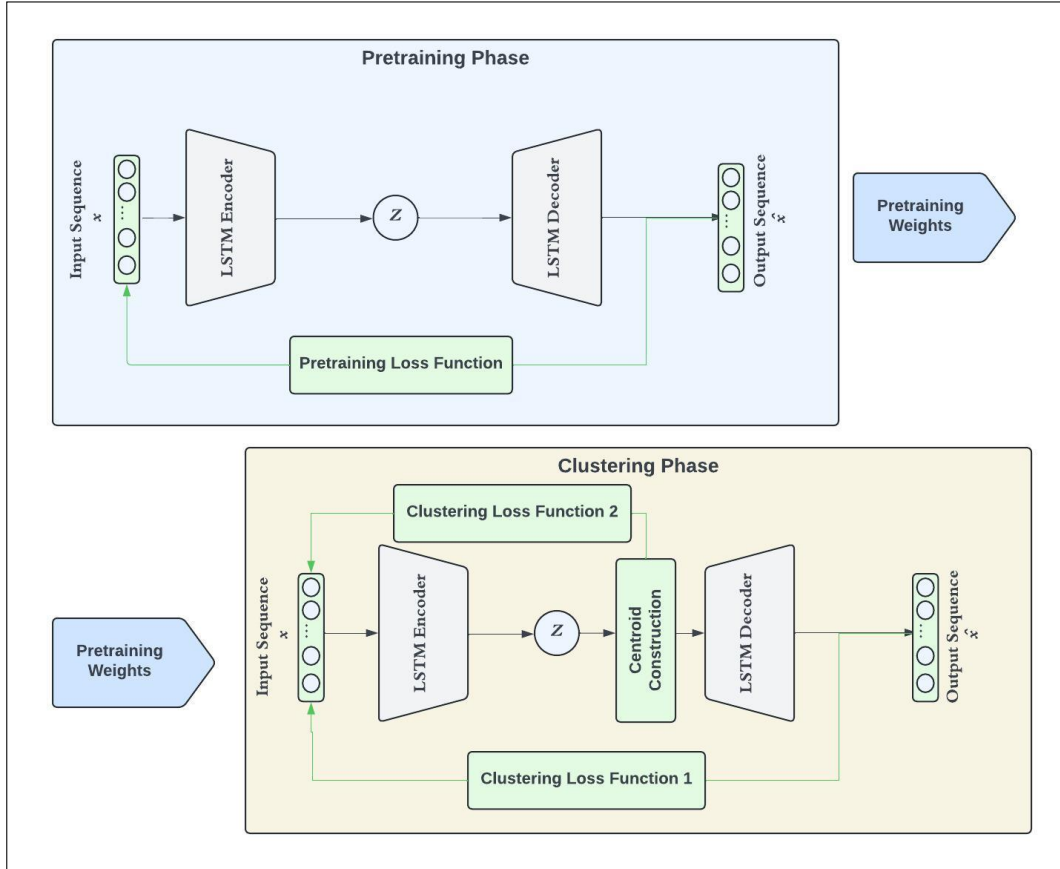


Figure 3.2: The Architecture of the Proposed LSTM-DynAE Model

3.3.1 LSTM-AE Architecture

An autoencoder (AE) is a special type of deep-forward neural network, typically utilized for representation learning and dimensionality reduction. The proposed model uses an LSTM-AE, a neural network built on an autoencoder architecture for encoding and decoding a sequence of input data, where an LSTM network is utilized to perform the encoding

and decoding process (Mobtahej, Zhang, Hamidi, & Zhang, 2022). LSTM network is a type of Recurrent Neural Network (RNN) designed especially for sequence problems. It can be thought of as the addition of loops to the feed-forward architecture of MultiLayer Perceptrons (MLP) Network. It is aware of the temporal structure in the inputs and able to model multiple parallel input series separately as it has recurrent connections so that the state from previous activations of the neuron from the previous time step is used as a context for formulating an output (Brownlee, 2017). Like other variations of autoencoders, the LSTM-AE consists of an LSTM-Encoder, a bottleneck, and an LSTM-Decoder. The architecture of LSTM-AE consists of the following two parts:

- **LSTM-Encoder:** The LSTM encoder $f_{\phi_e}(x_i)$ has an LSTM-encoder input layer and two LSTM-encoder layers. the first LSTM with 64 dimension, while the second layer is of 128 dimension. It performs mapping of data to the bottleneck layer. The mapping of the encoded data is mathematically represented as $z_i = f_{\phi_e}(x_i) \in \mathbb{R}^d$.
- **LSTM-Decoder:** The LSTM decoder $g_{\phi_d}(z_i)$ is a mirror of the encoder, with two LSTM layers. The first layer is of 128 dimensions, and the second layer is of 64 dimension. The extracted features are decoded and reconstructed according to the following mathematical representation, $\hat{x}_i = g_{\phi_d}(z_i) \in \mathbb{R}^d$.

The encoder and decoder dimension sizes (64, 128) were chosen empirically to balance reconstruction accuracy and computational efficiency. The aim of using LSTM-AE is to find a code for each input sample by minimizing the mean squared errors (*MSE*) between its input and output over all samples (Xie, Girshick, & Farhadi, 2015). Mean of Square Errors (*MSE*) present the reconstruction objective function for a set of data samples $x = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, as shown in Equation 3.15. Thus, the LSTM-AE is utilized to perform the initialization for the clustering in the pretraining phase.

$$\min L_1 = \min \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \quad (3.15)$$

3.3.2 Phase-I: Pre-training Phase

Pretraining is conducted as a distinct phase of clustering. Since Deep Embedded Clustering relies on the features extracted from the latent representation during pretraining, the feature extraction process plays a crucial role in determining the reliability of the model.

Consequently, the architecture of the autoencoder and the objective function employed during pretraining are key factors that influence this reliability. The AE architecture has been described previously, while the objective function for pretraining will be detailed in this section.

The standard reconstruction objective function (Guo, Zhu, Liu, & Yin, 2018) optimized the reconstruction process through data augmentation, yielding competitive results. Furthermore, studies (Ren, Wang, Li, & Xu, 2020) and (Hu, Miyato, Tokui, Matsumoto, & Sugiyama, 2017) demonstrated that data augmentation enhances prior knowledge in unsupervised learning. This technique involves rotating and rescaling data serving as a regularizer that improves generalization during the pretraining phase.

Conversely, (Berthelot, Raffel, Roy, & Goodfellow, 2018a) successfully enhanced interpolation in autoencoders (AEs) by incorporating an adversarial regularizer. Notably, ADEC and DynAE are innovative models that adapted reconstruction loss with adversarially constrained interpolation and data augmentation, resulting in improved outcomes (Mrabah et al., 2019). To achieve this competitive performance, the proposed reconstruction objective function of LSTM-DynAE is optimized with both an adversarial regularizer and data augmentation.

Figure 3.3 shows the full pretraining phase of the LSTM-DynAE model. The adversarial regularizer framework utilized in this proposed method is derived from the Adversarially Constrained Autoencoder Interpolation (ACAI) framework (Berthelot et al., 2018a). This framework comprises an autoencoder (AE) and a critic network, where the AE generates samples based on the input data. These samples embody a blend of semantic features.

Following data transformation operations, such as rescaling and rotation for data augmentation, let x_α represent the reconstructed data vector as indicated in Equation 3.17. In each training iteration t , two randomly selecting coefficients α_1 and α_2 from the range $[0, 1]$ are selected to determine x_α , the embedded representation data vectors x_1 and x_2 . Let $z_1 = f_{\phi_e}(x_1)$ and $z_2 = f_{\phi_e}(x_2)$ represent the latent codes of the two samples x_1 and x_2 , respectively. The interpolation of the latent codes z_1 and z_2 is denoted by Equation 3.16:

$$z_\alpha = \tau_1 f_{\phi_e}(x_1) + (1 - \tau_1) f_{\phi_e}(x_2) \quad (3.16)$$

From the interpolation of the latent spaces z_1 and z_2 , we compute $\hat{x}_\alpha = g_{\phi_d}(z_\alpha)$ which represents the reconstructed output of z_1 and z_2 as shown in Equation 3.17 for the reconstruction loss function.

The critic network, represented as C with learnable parameters ϕ_C , is trained to predict

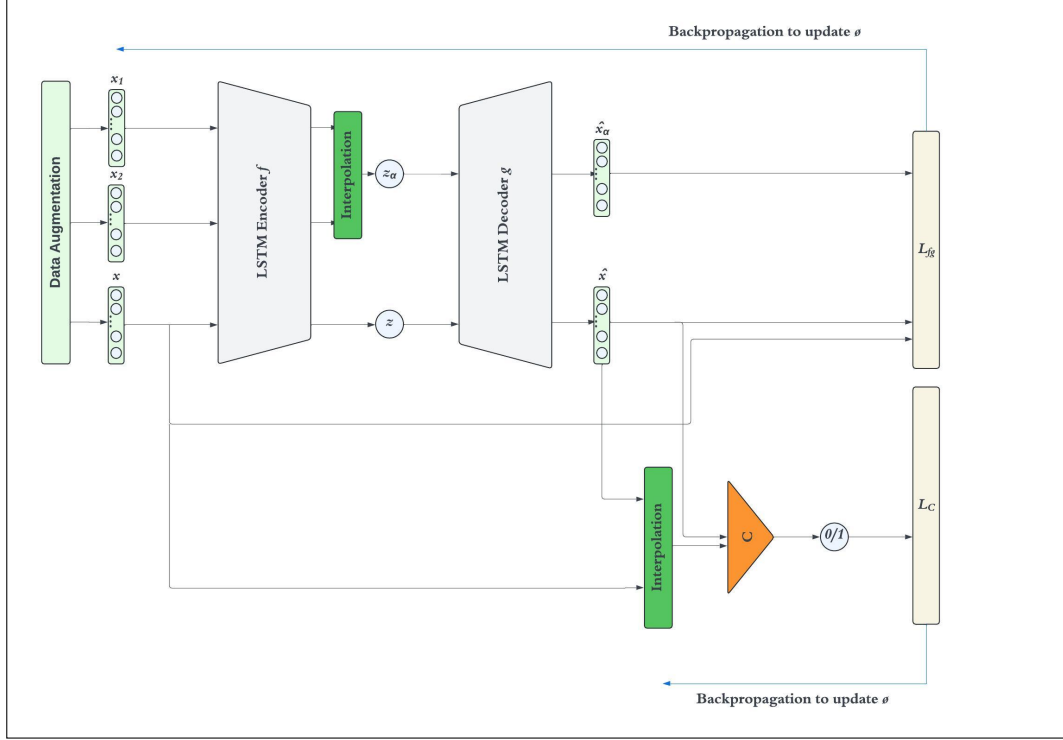


Figure 3.3: The Pretraining Phase of LSTM-DynAE Model

the interpolation coefficient τ_1 from input $\hat{x}_\alpha$. Concurrently, the main network is trained to reconstruct the input data in a way that deceives the critic network into believing that the interpolated outputs x_α are realistic. The loss functions for the auto-encoder, denoted as L_{fg} , and the critic, denoted as L_C , are articulated in Equations 3.18 and 3.19, respectively. The coefficient $\lambda \in [0, 1]$ is used to balance the loss functions. This regularization technique improves the realism of the interpolants, resulting in a smoother latent space (Mrabah, Khan, Ksantini, & Lachiri, 2020).

$$x_\alpha = g_{\phi_d}(\tau_1 f_{\phi_e}(x_1) + (1 - \tau_1) f_{\phi_e}(x_2)) \quad (3.17)$$

$$L_{fg}(\phi_e(t), \phi_d(t)) = \|x - \hat{x}\|_2^2 + \lambda \|\mathcal{C}(\hat{x}_\alpha)\|_2^2 \quad (3.18)$$

$$L_C(\phi_C(t)) = \|C(\hat{x}_\alpha) - \alpha\|_2^2 + \|\mathcal{C}(\tau_2 x + (1 - \tau_2)\hat{x})\|_2^2 \quad (3.19)$$

3.3.3 Phase-II: Clustering Phase

The clustering phase of LSTM-DynAE takes place after the pretraining phase, allowing for the fine-tuning of the autoencoder weights using a dynamic loss. As shown in Figure 3.4 this phase involves three key aspects: initializing the embedded clustering centers, defining the clustering loss function, and optimizing the loss function.

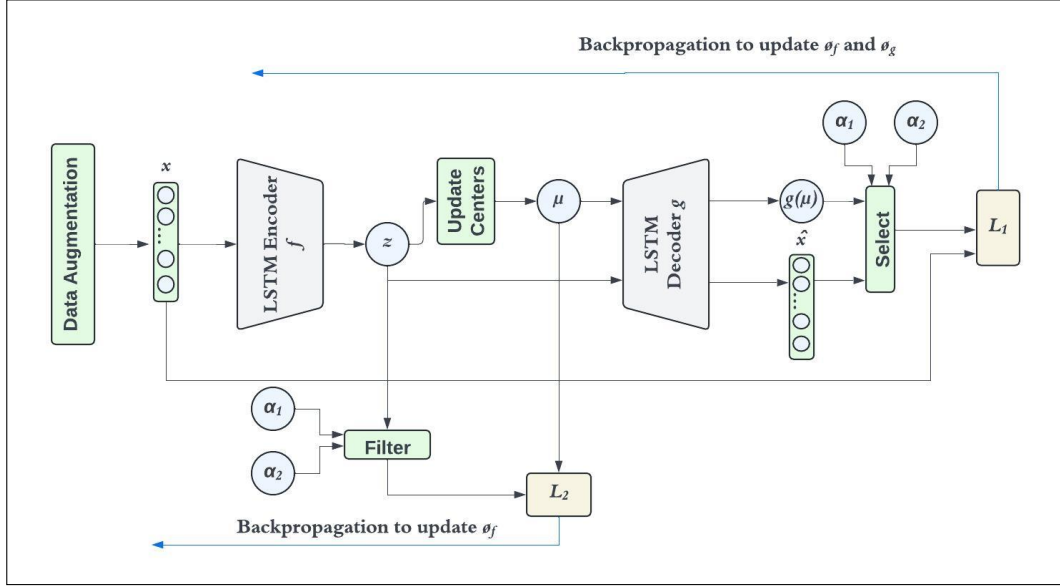


Figure 3.4: The Clustering Phase of LSTM-DynAE Model

3.3.3.1 The Initialization of Clustering Centers

K-means is a widely used clustering algorithm applied across various domains (Jabi, Pedersoli, Mitiche, & Ayed, 2019). For the sake of simplicity, K-means is employed in LSTM-DynAE to initialize the embedded clustering centers. While we didn't explicitly explore other initialization strategies in this thesis, methods like K-means++ or more advanced techniques that leverage density information could potentially offer further improvements by providing more robust initial centroids, especially for complex data distributions. This would be a valuable avenue for future work.

3.3.3.2 Dynamic Clustering Loss Function

Autoencoder based Deep Embedded Clustering methods involve two components in the clustering cost function. The first component addresses the reconstruction cost, while the second serves as the clustering objective function. This process involves the output of

LSTM-DynAE being the centroid vector for each sample. The decoder then takes on the task of generating the centers of these vectors. Subsequently, this output is fed back into the network as a supervisory signal. It is essential to maintain a balance between Feature Randomness and Feature Drift. This is important because the abrupt removal of the self-supervised loss reduces the reconstruction loss, leading to increased Feature Randomness. The dynamic loss ensures a gradual and smooth reduction of the reconstruction loss in favor of centroid construction.

After the pretraining phase, some of the embedded data points z_i may exhibit their highest clustering assignments q_{ir} due to their proximity to one another. This closeness renders them unreliable, making it difficult to accurately predict the corresponding centers for these samples, as they are situated near the boundaries of the clusters. These samples are referred to as conflicted data points. The dynamic loss function incorporates a condition that regulates the acceptable level of uncertainty to identify unconflicted data points during each training iteration. Let $\bar{S}$ represent the set of conflicted data points. The hyperparameters α_1 and α_2 fall within the interval $[0, 1]$. α_1 sets the minimum confidence threshold for the conflicted data points, while α_2 determines the minimum difference between the highest and second-highest assignment probabilities. Equation 3.20 illustrates $\bar{S}$, where p_{i1} denotes the highest assignment probability and p_{i2} represents the second-highest assignment probability. Thus, α_1 and α_2 serve as trade-off hyperparameters, as a conflicted sample must either have its highest assignment probability below α_1 or the difference between its highest and second-highest assignments must be less than the threshold α_2 .

$$\bar{S} = \{x_i \in X \mid p_{i1} < \alpha_1 \text{ or } (p_{i1} - p_{i2}) < \alpha_2\} \quad (3.20)$$

Based on the aforementioned points, reliable samples will be selected for centroid construction. This is accomplished using the first part of the dynamic loss function, as illustrated in Equation 3.21. There are two possible training strategies to achieve the confidence level at which the conflicted data points correspond to the most reliable centers; either through reconstruction or centroid construction. Consequently, the reconstruction cost is utilized until this confidence level is reached (Mrabah et al., 2019).

$$L_1(\phi_e(t), \phi_d(t)) = \sum_{i=1}^N \begin{cases} \|x_i - \hat{x}_i\|_2^2 & \text{if } x_i \in \bar{S} \\ \|g(\sigma(x_i)) - \hat{x}_i\|_2^2 & \text{otherwise.} \end{cases} \quad (3.21)$$

However, the selected samples for centroid construction do not accurately represent real data points. Therefore, the first nearest neighbor $(1 - NN)$, where NN represents the

nearest neighbor (Wettschereck & Dietterich, 1995), is employed for each embedded center (μ'_r) as a substitute for the generated centroids. This indicates that the hyperparameters may vary depending on the dataset used for model training. The hyperparameters (α_1) and (α_2) are calculated using Equation 3.22, where (κ) is the confidence threshold that determines (α_1) and (α_2).

$$\alpha_1 = \frac{\kappa}{K} \quad \text{and} \quad \alpha_2 = \frac{\alpha_1}{2}, \quad \kappa \in [1, K] \quad (3.22)$$

Equation 3.23 outlines the process for tuning the hyperparameters, with τ_p representing the training progress. This strategy ensures that during the initialization phase, more emphasis is placed on reconstruction rather than centroid construction, which may result in slower convergence with the dynamic loss function. The model is trained until reconstruction is completely eliminated. This formulation creates a natural dynamic balance between the two learning objectives. The role of the self-supervision task (reconstruction) is effectively weighted by p , which decreases over the course of training. Conversely, the role of the pseudo-supervision task (clustering and centroid construction) is weighted by $(1-p)$, which progressively increases. This smooth transition is the core mechanism that mitigates FR early in training and counters FD later in training.

$$\tau_p = \frac{\text{no. of reconstructed samples}}{N} = \frac{|\bar{S}|}{N} \quad (3.23)$$

Equation 3.21 depicts the clustering loss function. However, minimizing L_1 could compromise the embedded features. To address this, L_2 was introduced in DynAE to ensure that the latent representations are conducive to centroid-based clustering. The second loss penalizes the distance between the data vectors and their corresponding centroids, as shown in Equation 3.24, where S represents the unconflicted data points, as detailed in Equation 3.25 (Mrabah, Khan, et al., 2020). The same loss functions from DynAE are utilized in the proposed LSTM-DynAE algorithm.

$$L_2(\phi_e(t), \phi_d(t)) = \sum_{x_i \in S} \|f_{\phi_e}(x_i) - \sigma(x_i)\|_2^2 \quad (3.24)$$

$$S = \{x_i \in X \mid p_{i1} \geq \alpha_1 \text{ and } (p_{i1} - p_{i2}) \geq \alpha_2\} \quad (3.25)$$

The complete dynamic objective function is presented in Equation 3.26. Similar to the pretraining phase, the loss function L is regularized through data augmentation. The approach of LSTM-DynAE does not require a balancing hyperparameter as DynAE which

aims to achieve a better trade-off between FR and FD by utilizing learning dynamics without the need for any balancing hyperparameter. This is because the weight of the pseudo-supervision component (clustering loss) is implicitly determined by the number of 'unconflicted' samples. As training progresses and the model becomes more confident, the number of unconflicted samples increases, and the clustering objective naturally gains more weight. Conversely, early in training, when many samples are conflicted, the reconstruction objective dominates. This adaptive shift, driven by the evolving confidence of the model, eliminates the need for a manually tuned balancing hyperparameter. The potential trade-off is that the initial learning might be slower as the model prioritizes reconstruction until sufficient confidence is built for clustering, but this leads to more robust final representations.

$$L = L_1 + L_2 \quad (3.26)$$

3.3.3.3 Optimization

The combined loss function of L_1 and L_2 is optimized using mini-batch gradient descent and backpropagation. Training is conducted for each iteration t until either the maximum number of iterations $MaxItr$, is reached or τ_p falls below a specified fraction of the *total* training set.

During the training of LSTM-DynAE, the dynamic objective function transitions smoothly and gradually from pure self-supervision to pure pseudo-supervision. At the outset, the model is pretrained with a strictly self-supervised objective function. As training progresses, the final objective function approaches a state of almost complete pseudo-supervision, with the weight of pseudo-supervision τ_p becoming very low. This shift incorporates transient objectives that blend self-supervision and pseudo-supervision strategies.

As the training continues, the role of pseudo-supervision increases while that of self-supervision diminishes. The rationale behind this approach can be summarized in several key points: initially, pseudo-supervision is unreliable due to a high number of fake labels, which results in Feature Randomness. While self-supervision is effective for learning general-purpose features, it does not align with the ultimate clustering objective. Additionally, the combination of pseudo-supervision and self-supervision can lead to Feature Drift. However, when reliable pseudo-labels are available, pseudo-supervision proves to be highly effective. This adaptive balance between self-supervision and pseudo-supervision ultimately enhances the clustering performance of LSTM-DynAE. For more details about

the effect of the dynamic objective function refer to (Mrabah et al., 2019).

Algorithm 1 and the flowchart shown in Figure 3.5 depict the deep embedded clustering process of the proposed LSTM-DynAE model, illustrating the steps of the pretraining and the clustering phases. The computational complexity of the proposed **LSTM-DynAE** model is approximately

$$\mathcal{C}(\text{LSTM-DynAE}) = O(mLH^2T),$$

where m denotes the number of training iterations, L the number of recurrent layers, H the maximum number of hidden units, and T the sequence length. This cost arises mainly from the recurrent forward and backward passes within each LSTM cell, which scale quadratically with the hidden size. Similar to **DEC**, **IDEC**, **DCN**, and **DynAE**, the proposed model maintains the same asymptotic computational order, implying that execution time comparisons are primarily determined by the number of iterations required for convergence. Mini-batch stochastic gradient descent (SGD) reduces both computation and memory cost by a factor of $\frac{N}{B}$, where N is the total number of samples and B is the batch size. Thus, **LSTM-DynAE** preserves the scalability of DynAE while extending its applicability to sequential data.

3.3.4 Evaluation Metrics

The performance of the deep-embedded clustering method has been assessed by the two common metrics, Accuracy (Acc) and Normalized Mutual Information (NMI) (Cai, He, & Han, 2011). Acc and NMI are allocated within the interval $[0, 1]$, where the higher value means a better performance. Equation 3.27 and Equation 3.28 represent the mathematical definition of Acc , and NMI , respectively. Both equations depend on, y_t and y_p , while y_t denotes the ground truth labels vector, and y_p is the vector of the predicted clustered labels. Meanwhile, I denotes the mutual information, H represents the entropy, and T is selected from the set of best mapping that matches the predicted clusters to their ground truth labels. The Hungarian Algorithm (Khan, 2020) demonstrates this mapping.

$$Acc(y_p, y_t) = \max_T \left\{ \frac{\sum_{i=1}^N 1 \{y_t(i) = T(y_p(i))\}}{N} \right\} \quad (3.27)$$

$$NMI(y_p, y_t) = \frac{I(y_p, y_t)}{\frac{1}{2}[H(y_t) + H(y_p)]} \quad (3.28)$$

Algorithm 1 Sequential LSTM Dynamic Autoencoder

- 1: **Input:** Input time series data: X , Number of Clusters: K , Maximum Iterations: $MaxItr$,
Confidence Threshold: κ , Learning Rate: ω , Convergence threshold: $total$
 - 2: **Output:** Autoencoder weights: w
 - 3: **Stage 1** $\rightarrow$ Pretraining Phase
 - 4: Perform LSTM Encoder $f_{\phi_e} : x \rightarrow z$
 - 5: Perform LSTM Decoder $g_{\phi_d} : z \rightarrow x$
 - 6: Initialize pretrain w via Equation 3.18 and Equation 3.19
 - 7: **Stage 2** $\rightarrow$ Clustering Phase
 - 8: Initialize embedded centroids η using K-means
 - 9: Compute α_1 and $\alpha_2 \leftarrow \kappa$ via Equation 3.22
 - 10: **while** list of conflicted points is not empty $\leftarrow |X|$ **do**
 - 11: **for** $i = 0$ to $MaxItr$ **do**
 - 12: Update list of conflicted points is not empty $\leftarrow (X, \eta, \alpha_1, \alpha_2)$
 - 13: Update Centroids $\eta \leftarrow (X, K)$
 - 14: Update α_1 and $\alpha_2 \leftarrow \kappa$
 - 15: **if** conflicted points $< total$ **then**
 - 16: End training
 - 17: **end if**
 - 18: Compute L via Equation 3.26
 - 19: $w \leftarrow w - \vartheta \frac{\partial L}{\partial w}$
 - 20: **end for**
 - 21: **end while**
-

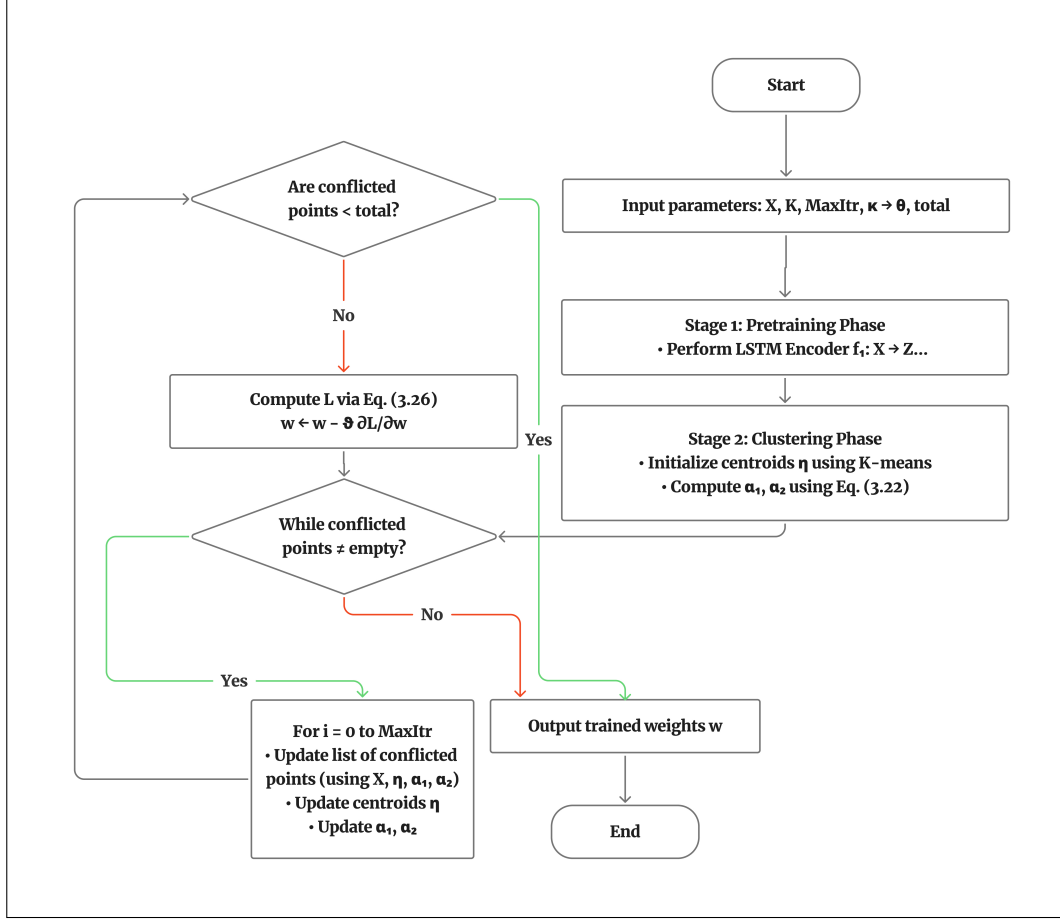


Figure 3.5: The Flowchart of the LSTM-DynAE Model

Two additional evaluation metrics, Feature Randomness(ΔFR) and Feature Drift (ΔFD) are used in the evaluation of the model presented in this study, as they are part of the research's contributions. The two metrics are within the interval $[-1, 1]$, where the higher values are better. The definition of (ΔFR) is the cosine of the angle between the derivatives of the supervised and unsupervised objective functions. The mathematical definition of ΔFR and ΔFD are presented by Equations 3.29 and 3.30, respectively, as borrowed from (Mrabah et al., 2019). The notation ω denotes the network parameters, and $y_{pretext}$ denotes the self-acquired labels for the pretext task.

$$\Delta FR = \cos \left(\frac{\partial L(x, y_t, \omega)}{\partial \omega}, \frac{\partial L(x, y_p, \omega)}{\partial \omega} \right) \quad (3.29)$$

$$\Delta FD = \cos \left(\frac{\partial L_P(x, y_p, \omega)}{\partial \omega}, \frac{\partial L_S(x, y_{pretext}, \omega)}{\partial \omega} \right) \quad (3.30)$$

3.4 Proposed ConvLSTM-DynAE Model for Load Profiling

Deep autoencoders have gained traction across various sectors within the energy domain, such as load monitoring (Verma, Singh, & Majumdar, 2021), forecasting (Y. Wang et al., 2020), and pattern clustering (Teeraratkul, O'Neill, & Lall, 2017). The proposed ConvLSTM-DynAE architecture and its dual phases are depicted in Figure 3.6. As illustrated in the Figure, the ConvLSTM-DynAE model is an enhancement of the previously introduced LSTM-DynAE model. The primary distinction lies in the architecture of the autoencoder, which has been upgraded to a ConvLSTM autoencoder. This modification improves the ability of the model to capture the spatial and temporal features of smart meter data, thereby enhancing load profiling.

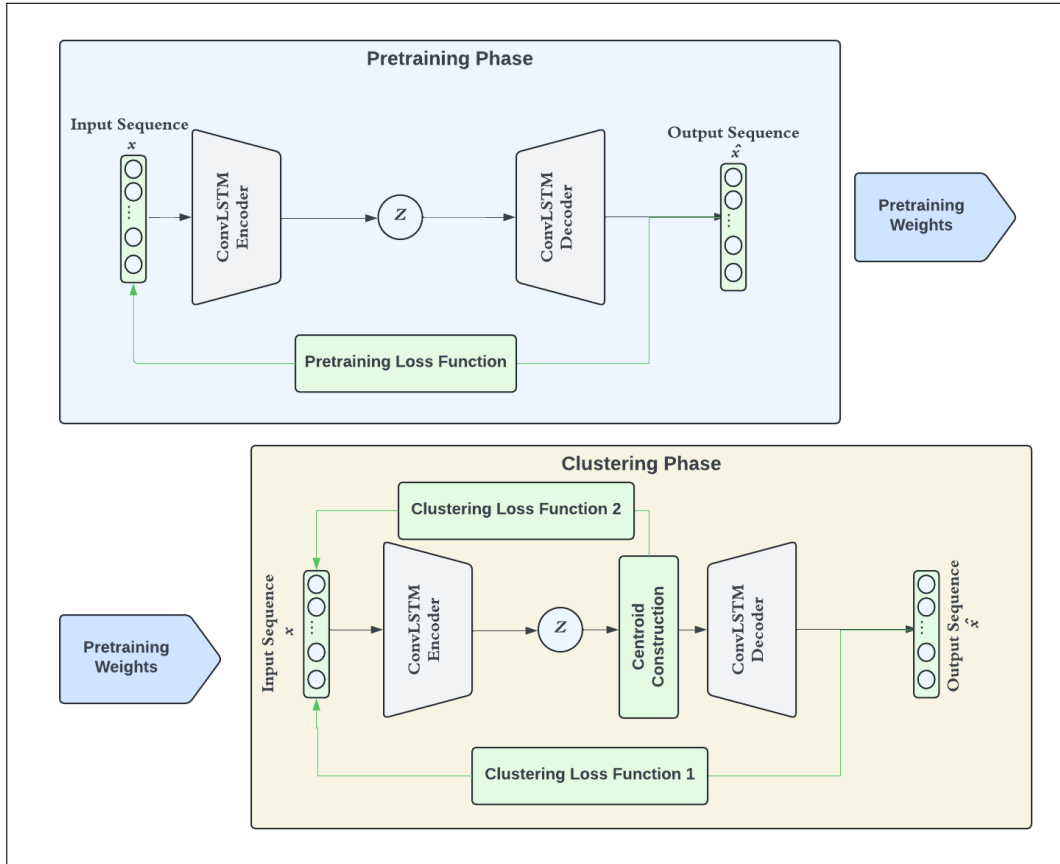


Figure 3.6: The Architecture of the proposed ConvLSTM-DynAE Model

The subsequent subsections detail the architecture of the ConvLSTM-AE utilized in the proposed model, followed by an overview of the pretraining and clustering phases,

which mirror the stages of the LSTM-DynAE model. Finally, a comprehensive explanation of the load profiling framework methodology will be provided.

3.4.1 ConvLSTM-AE Architecture

Figure 3.7 shows a diagram of the autoencoder architecture with hyperparameters (units, filters, activation functions, and padding). The proposed ConvLSTM-AE leverages convolutional LSTM layers to capture both spatial and temporal dependencies within smart meter data. The encoder extracts temporal features through stacked ConvLSTM2D layers followed by normalization and dropout to stabilize learning and mitigate overfitting. The decoder mirrors this design, reconstructing the original input through ConvLSTM2D layers and a time-distributed dense layer. A detailed summary of all parameter values (number of units, kernel size, activation functions, and dropout ratios) used in the implementation is provided in Section 4.2 (Implementation Settings).

The mathematical representations of the convolutional LSTM encoder and decoder mappings are given by the previously defined equations $z_i = f_{\phi_e}(x_i) \in \mathbb{R}^d$, and $\hat{x}_i = g_{\phi_d}(z_i) \in \mathbb{R}^d$, respectively.

3.4.2 Phase-I: Pretraining Phase

The initialization for clustering is carried out during the pretraining phase using the proposed ConvLSTM autoencoder architecture. As illustrated in Figure 3.8 shows the full pretraining phase of the ConvLSTM-DynAE model, the pretraining phase is a mirror of the pretraining performed in LSTM-DynAE model, with the use of the ConvLSTM-AE. For details refer to subSection 3.3.2

3.4.3 Phase-II: Clustering Phase

The clustering phase of ConvLSTM-DynAE takes place after the pretraining phase, allowing for the fine-tuning of the autoencoder weights using a dynamic loss. This phase involves three key aspects: initializing the embedded clustering centers, defining the clustering loss function, and optimizing the loss function. As shown in Figure 3.9 the clustering phase of ConvLSTM-DynAE is similar to the clustering phase of LSTM-DynAE integrated with dynamic loss function. For details refer to the subSection 3.3.3.

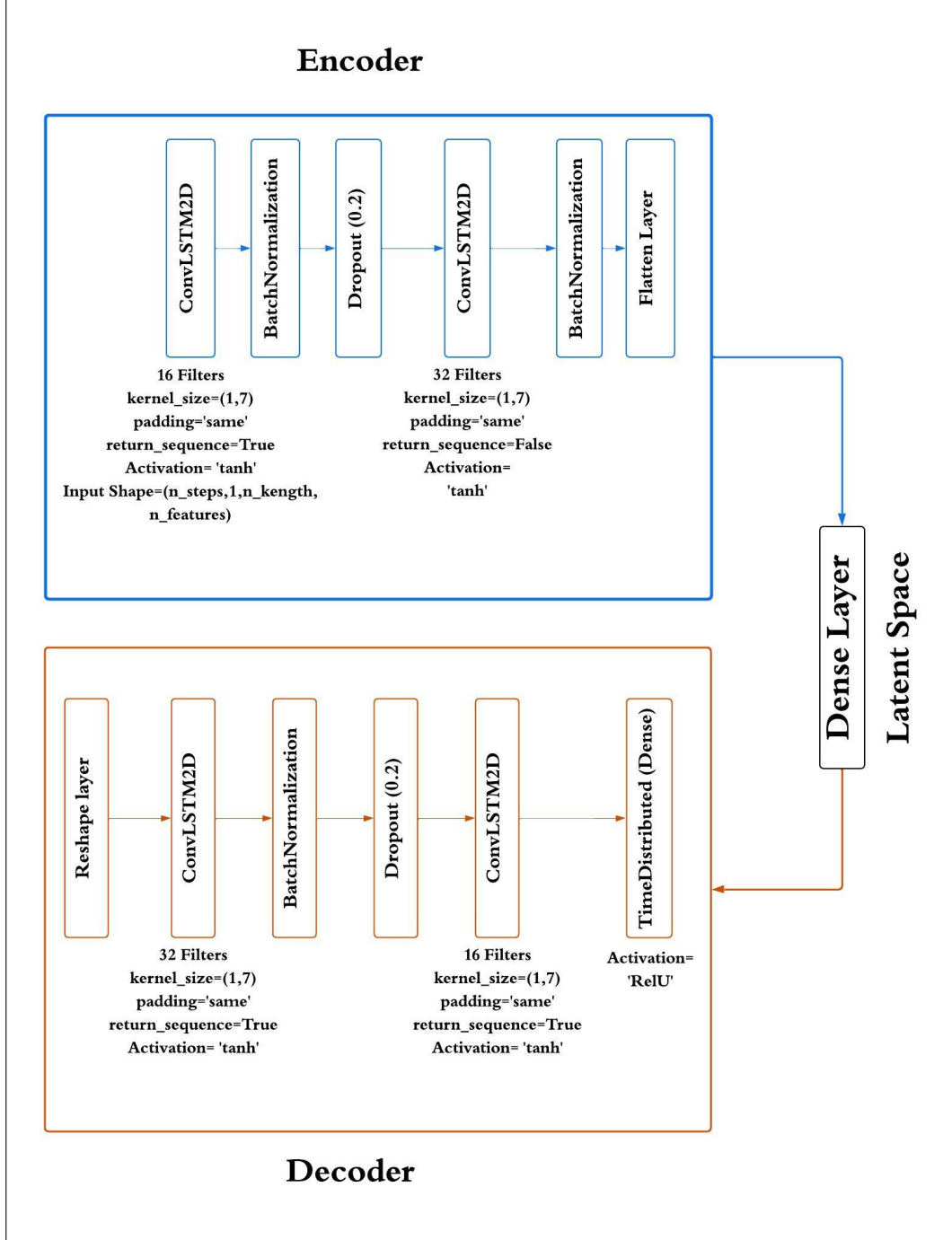


Figure 3.7: Architecture of Convolutional LSTM Autoencoder

Algorithm 2 and the flowchart in Figure ?? illustrate the deep embedded clustering process of the proposed ConvLSTM-DynAE model, illustrating the steps of the pretraining and the clustering phases. Excluding the pretraining phase, the overall computational

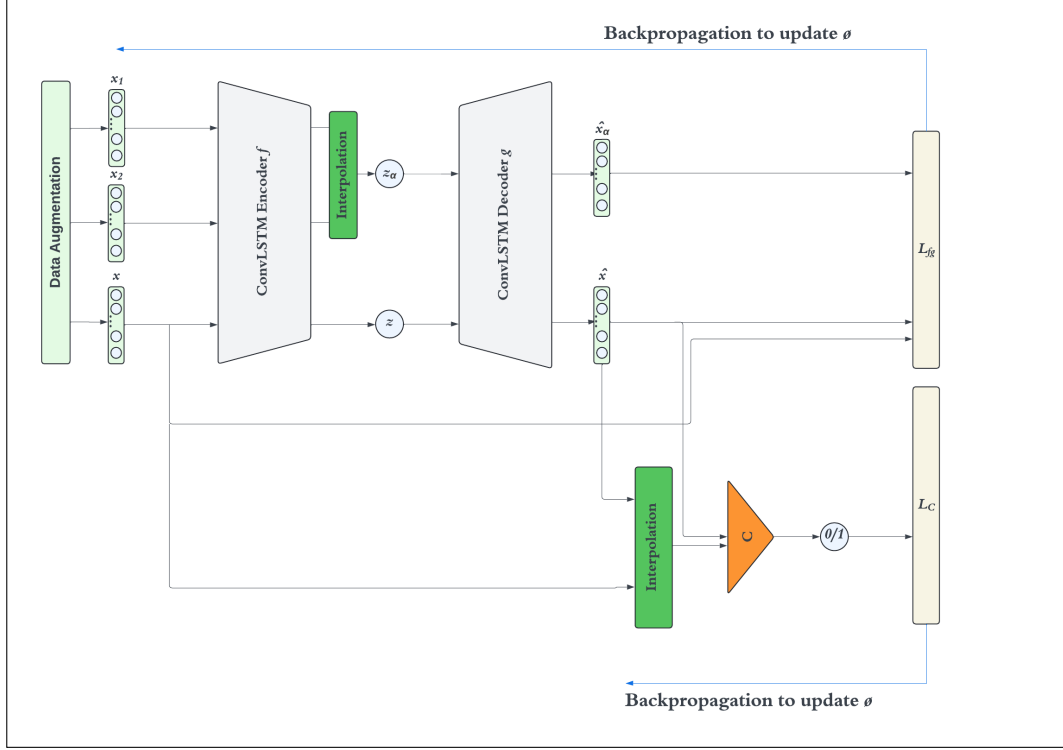


Figure 3.8: The Pretraining Phase of ConvLSTM-DynAE Model

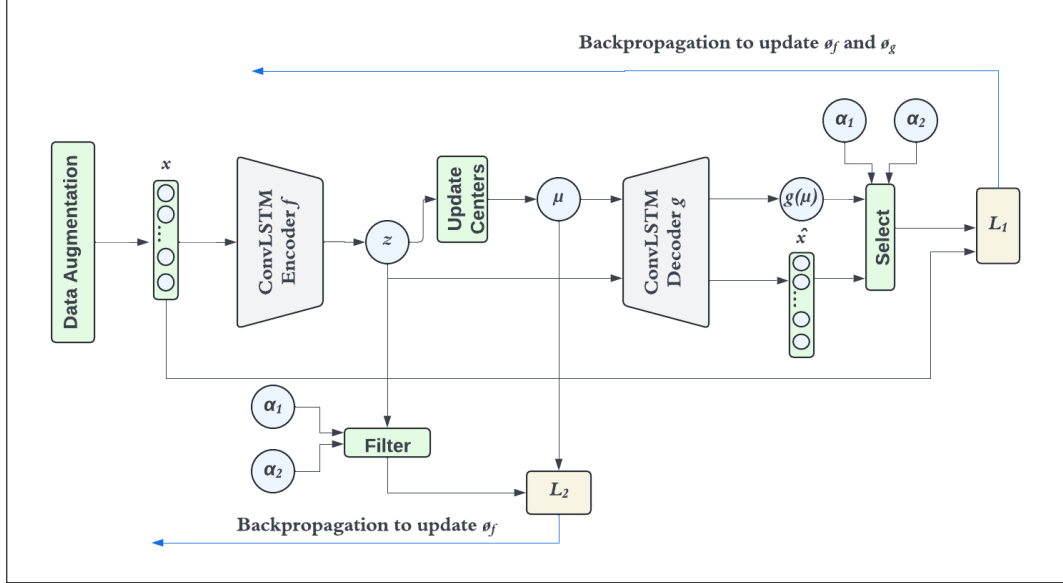


Figure 3.9: The Clustering Phase of ConvLSTM-DynAE Model

complexity of **ConvLSTM-DynAE** is given by

$$\mathcal{C}(\text{ConvLSTM-DynAE}) = O(mLH^2k^2T),$$

where k represents the convolution kernel size. The convolutional operations introduce an additional constant factor k^2 , yet they are executed on compact latent feature maps, mitigating runtime impact. Consequently, **ConvLSTM-DynAE**, **LSTM-DynAE**, and **DynAE** share the same asymptotic order of complexity. In practice, ConvLSTM's weight-sharing and spatiotemporal locality typically reduce the number of iterations m required to reach convergence, providing superior efficiency despite similar theoretical cost.

Algorithm 2 Convolutional Sequential LSTM Dynamic Autoencoder

```

1: Input: Input time series data:  $X$ , Number of Clusters:  $K$ , Maximum Iterations:  $MaxItr$ ,
   Confidence Threshold:  $\kappa$ , Learning Rate:  $\omega$ , Convergence threshold:  $total$ 
2: Output: Autoencoder weights:  $w$ 
3: Stage 1  $\rightarrow$  Pretraining Phase
4: Perform Convolutional LSTM Encoder  $f_{\phi_e} : x \rightarrow z$ 
5: Perform Convolutional LSTM Decoder  $g_{\phi_d} : z \rightarrow x$ 
6: Initialize pretrain  $w$  via Equation 3.18 and Equation 3.19
7: Stage 2  $\rightarrow$  Clustering Phase
8: Initialize embedded centroids  $\eta$  using K-means
9: Compute  $\alpha_1$  and  $\alpha_2 \leftarrow \kappa$  via Equation 3.22
10: while list of conflicted points is not empty  $\leftarrow |X|$  do
11:   for  $i = 0$  to  $MaxItr$  do
12:     Update list of conflicted points is not empty  $\leftarrow (X, \eta, \alpha_1, \alpha_2)$ 
13:     Update Centroids  $\eta \leftarrow (X, K)$ 
14:     Update  $\alpha_1$  and  $\alpha_2 \leftarrow \kappa$ 
15:     if conflicted points  $< total$  then
16:       End training
17:     end if
18:     Compute  $L$  via Equation 3.26
19:      $w \leftarrow w - \vartheta \frac{\partial L}{\partial w}$ 
20:   end for
21: end while

```

3.4.4 Load Profiling Framework

In this section, we present the methodology of the load profiling framework which is enhanced with ConvLSTM-DynAE clustering model. The framework comprises three main

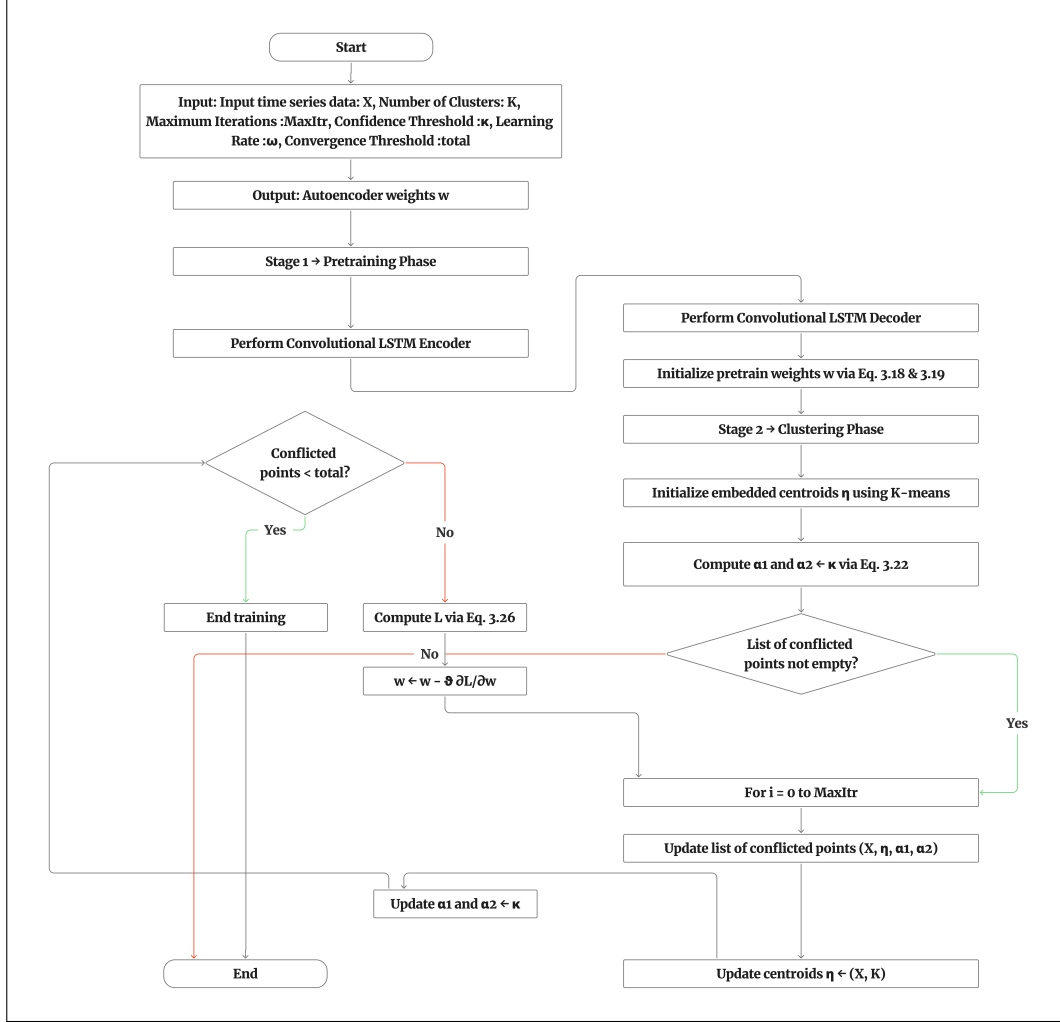


Figure 3.10: The Flowchart of the ConvLSTM-DynAE Model

steps: data preprocessing, clustering process, and the load curves generation. Each step is elaborated upon in the subsequent subsections. The load profile framework is inspired from the work published by (Eskandarnia et al., 2022), but in our study the clustering process of time series smart meter data is performed by the proposed ConvLSTM-DynAE model, which is presented in the previous section. The steps of the load profiling framework is presented in the following subsections:

3.4.4.1 Step-1: Data Preprocessing

A smart meter dataset can be represented as a matrix with $a \times q$ dimensions, where a denotes the number of readings taken over time and q represents the number of households. It is important to note that, despite the data being preprocessed, the framework circumvents

many complex preprocessing tasks, such as feature selection and dimensionality reduction, as previously mentioned. Initially, the dataset is typically resampled, and the frequency of the time series is reduced (i.e., downsampling) based on the specific requirements of the task. The next step in the data preprocessing phase is data imputation. During this stage, if a household has missing consumption readings for x instances and x is below a certain defined threshold, that household will be excluded from the analysis; this threshold is determined based on the value of q .

The formula presented in Equation 3.31 is employed to address missing values. As indicated in Equation 3.31, the readings from the same time and date in the following year (or the previous year) are utilized to replace the null values. If those readings are also unavailable, the missing values are filled with the average consumption for the same month. The reading of the house consumption for the next year or previous year is added to 365 (the total days of a year) and multiplied by 48, as the readings are of half-hourly intervals.

$$x_i = \begin{cases} x_{i+365 \times 48} & \text{if } x_{i+365 \times 48} \neq \text{null} \\ x_{i-365 \times 48} & \text{if } x_{i-365 \times 48} \neq \text{null} \\ \frac{\sum_{b=1}^{30 \times 48} x_b}{30 \times 48} & \text{otherwise for } x_b, x_i \text{ of the same month} \end{cases} \quad (3.31)$$

The next step involves data normalization, as outlined in Equation 3.32, which ensures that each consumption reading falls within the range of $0 \leq x \leq 1$. Min-Max normalization is a widely used technique for scaling data in preparation for training deep neural networks. Initially, the minimum $\min(x_i)$ and maximum $\max(x_i)$ values for each timestamp are determined, after which Equation 3.32 is applied to all households in the dataset.

$$x_{ji} = \frac{x_{ji} - \min(x_i)}{\max(x_i) - \min(x_i)} \quad \forall i = \{1, \dots, a\} \text{ and } \forall j = \{1, \dots, q\} \quad (3.32)$$

The dataset was further pre-processed. For instance, the dataset is typically re-sampled, and the frequency of the time series is reduced (i.e., down-sampling). Additionally, data reshaping will be carried out, where the smart meter data is rotated or restructured before being input into the algorithm. This process results in a matrix of size $b \times a$, with each row in the reshaped matrix representing the consumption readings for a particular household. Moreover, according to the architecture of the autoencoder used, the data input should be reshaped to be suitable for reading by the input layer. Since the autoencoder used in this study is convolutional LSTM autoencoder, the two dimensional data is transformed into spatiotemporal sequences (5D tensor). Each data input sample is reshaped into (*samples*,

timesteps, height, width, channels). This reshaping is applied only for the input of the Convolutional LSTM autoencoder for the proposed clustering method.

3.4.4.2 Step-2:Clustering

The proposed ConvLSTM-DynAE clustering method will be utilized in the load profiling framework to examine its effect in enhancing the load profiling. The proposed clustering model is discussed separately in Section 4. Similar to other deep clustering methods, DEC, IDEC, and DynAE, our algorithm requires the number of clusters k to be provided as input. To determine the optimal number of clusters, the algorithm will conduct a grid search for k and compute the Silhouette Coefficient (Sil) (Rousseeuw, 1987). The best value of k will then be selected based on the results obtained.

The Clustering Algorithm: The proposed ConvLSTM-DynAE algorithm (see Algorithm 2) is adopted to perform the clustering process of the smart meter data. The algorithm as mentioned is a double-stage deep clustering approach, where the first stage (*i.e., pretraining*) is initialized by a convolutional LSTM autoencoder, and the second stage (*i.e., clustering*) is performed and optimized with the dynamic clustering loss function presented by Equation 3.26 to mitigate the trade-off between FR and FD as explained in the section of the proposed model.

Assessment of Clustering Performance: Since the data from smart meters is unlabeled, evaluating the quality of the clustering results can be difficult (Vercamer, Steurtewagen, Van den Poel, & Vermeulen, 2015). Cluster Validity Indices (CVIs) are frequently employed for assessing and validating load profiling tasks, especially when the number of clusters and other clustering parameters are not known. Numerous CVIs have been proposed and evaluated for clustering tasks in general (Kim & Ramakrishna, 2005). Among these, the Silhouette Validity Index is particularly popular (Sun et al., 2020; Y. Wang et al., 2015). The Silhouette coefficient is calculated using the formula presented in Equation 3.33:

$$Sil = \frac{\gamma - \beta}{\max(\beta, \gamma)} \quad (3.33)$$

Where γ represents the distance between a sample and the nearest cluster to which it does not belong, and β is the average distance within the cluster. The Silhouette (Sil) score combines measures of compactness and separation to calculate its value (Rousseeuw, 1987). This calculation is designed to ensure that the SH score consistently falls within the range of $[-1, 1]$. Sil seeks to maximize the difference $(\gamma - \beta)$, resulting in a score that

reflects the quality of clustering. However, in load profiling tasks, as in other real-world clustering situations without a known ground truth, identifying the optimal number of clusters becomes more challenging. Breakpoints on the CVI curve, known as the Knee Technique, are often considered effective solutions to this issue (Zou, Li, Yang, Bai, & Zheng, 2017). In this work, a breakpoint in the Silhouette coefficient is utilized to determine the best number of clusters (Sun et al., 2020; S.-L. Yang, Shen, & Zhaou, 2013).

To obtain evaluation and benchmarking results, three deep clustering models were chosen, DEC, IDEC, and DynAE. Moreover, five classical clustering algorithms were chosen: K-means, Agglomerative Clustering (Hierarchical), DBSCAN, Spectral, and Gaussian Mixture Model (GMM). A preprocessed matrix was input into these algorithms. The hyperparameters for each algorithm were optimized, and the best Silhouette Index (Sil) for each technique was documented in the results.

3.4.4.3 Step-3: Generating Load Profiles

This step involves calculating the centroid for each cluster within the $q \times a$ space. To achieve this, a cluster matrix M is derived from the previous stage and combined with the smart meter readings D . Subsequently, a matrix T of size $a \times M$ is populated, where for each of the a readings, the consumption for cluster k is determined as the average consumption of all households within that cluster. The outcome is k univariate time series that represent the load profile for each cluster. To assess the quality of these load profiles, the resulting time series are reshaped and visually displayed.

Algorithm 3 and the flowchart in Figure 3.11 details the complete load profiling algorithm proposed for smart meter data. Given q a time series of size $a \times 1$, where a represents the number of time stamps and q denotes the number of households, each time series i contains the consumption data for house i at each timestamp. The proposed algorithm will categorize the houses into classes $C_1, C_2, \dots, C_k$, ensuring that the clusters are mutually exclusive. Algorithm 2 effectively clusters smart meter readings to identify patterns in energy consumption and generates load profiles for each cluster. By using the Silhouette Coefficient (Sil), it ensures that the chosen number of clusters provides meaningful insights into the data.

When the computational complexities of the clustering models are computed, the results show that both models share the same asymptotic order of complexity as existing deep clustering algorithms such as **DEC**, **IDEC**, **DCN**, and **DynAE**. The additional convolutional factor k^2 in ConvLSTM-DynAE introduces only a constant overhead, as convo-

lutional operations are performed on compact latent feature maps. Hence, execution-time differences among these methods mainly depend on the number of iterations required for convergence rather than theoretical order. In practice, **LSTM-DynAE** and **ConvLSTM-DynAE** achieve faster convergence and higher clustering quality because they integrate temporal and spatiotemporal modeling with a dynamic loss mechanism that gradually transitions from reconstruction to clustering. This dynamic adaptation effectively mitigates Feature Randomness (FR) and Feature Drift (FD), leading to more stable and discriminative latent representations. Furthermore, by leveraging mini-batch stochastic gradient descent, the proposed models maintain linear scalability with dataset size while providing superior performance in accuracy, NMI, and Silhouette scores. Consequently, although they share the same theoretical complexity class as previous methods, **LSTM-DynAE** and **ConvLSTM-DynAE** demonstrate better computational efficiency and improved clustering robustness due to faster convergence, reduced noise sensitivity, and enhanced representation stability.

Algorithm 3 The Framework of Load Profiling

```
1: Input: Smart meter readings  $B$ , Number of timepoints  $a$ , Features  $q$ 
2: Output: Load profiles matrix  $T$ , Cluster labels  $M$ 
3: Step 1  $\rightarrow$  Preprocessing
4:  $D \leftarrow \text{Preprocessing}(B)$   $\triangleright B \leftarrow q \times a$  matrix
5:  $B \leftarrow D$   $\triangleright$  Raw data replacement
6: Step 2  $\rightarrow$  Clustering & Analysis
7: for  $j \leftarrow 2$  to 10 do
8:    $k \leftarrow j$ 
9:   Perform Deep Clustering( $D, k$ ) using Algorithm 2
10:  Register Silhouette Coefficient  $Sil_j \leftarrow \text{Equation 3.33}$ 
11: end for
12: Visualize Silhouette Index  $Sil \leftarrow \{Sil_2, \dots, Sil_{10}\}$ 
13: Select optimal  $k \leftarrow Sil$ 
14:  $K \leftarrow \text{ClusterLabels}(D, k)$   $\triangleright M \leftarrow n \times 1$  vector
15: Step 3  $\rightarrow$  Load Profile Generation
16:  $L \leftarrow (D \parallel M)$   $\triangleright$  Matrix concatenation
17: Initialize  $T \leftarrow a \times k$  matrix
18: for  $l \leftarrow 1$  to  $k$  do
19:   for  $i \leftarrow 1$  to  $a$  and  $j \leftarrow 1$  to  $q$  do
20:      $T_{ij} \leftarrow \text{Mean}(L_{ji})$  where  $L[a+1] \leftarrow l$ 
21:   end for
22: end for
23: Step 4  $\rightarrow$  Evaluation & Visualization
24: for  $m \leftarrow 1$  to  $k$  do
25:    $\text{Profile}_m \leftarrow T[m]$   $\triangleright$  Cluster-specific profile
26:    $\text{DailyProfile}_m \leftarrow \text{Resample}(T[m])$ 
27:   Visualize  $\text{DailyProfile}_m$ 
28: end for
```

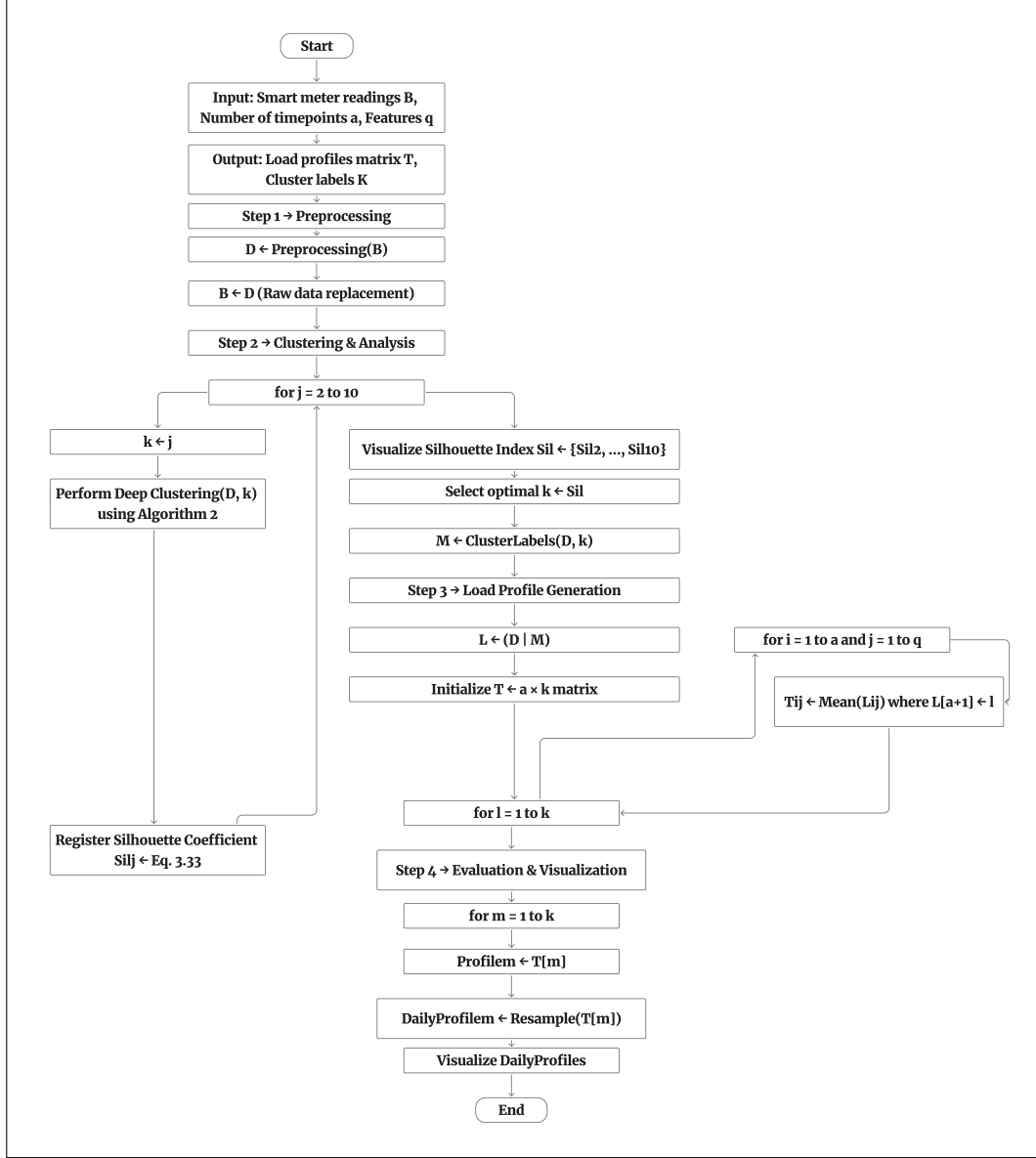


Figure 3.11: The Flowchart of the Load Profiling Framework

3.5 Summary

This chapter outlined the methodology employed in the study, concentrating on two deep embedded clustering models: the LSTM-DynAE and the ConvLSTM-DynAE. It started with the LSTM-DynAE model, which comprised an LSTM autoencoder, a pretraining phase for feature extraction, and a clustering phase that applied a dynamic loss function to improve clustering effectiveness. The chapter then introduced the ConvLSTM-DynAE model, which enhanced the LSTM-DynAE by utilizing a ConvLSTM autoencoder to more

effectively capture spatial and temporal features in smart meter data and perform load profiling. The methodology of load profiling framework presented including data preprocessing, clustering by the ConvLSTM-DynAE model, and load profile generation through the calculation of average consumption for each cluster. In summary, the chapter provided a clear overview of the methodologies for developing and implementing the LSTM-DynAE and ConvLSTM-DynAE models.

Chapter 4

Results and Discussions

4.1 Overview

This chapter showcases the experimental results derived from the two proposed models: LSTM-DynAE and ConvLSTM-DynAE. It is structured into three sections. Section 4.2 provides the implementation settings of the experiments for each model. Section 4.3 shows a comprehensive overview of the outcomes of the LSTM-DynAE model, including the baselines, datasets, and results. Section 4.4 concentrates on the ConvLSTM-DynAE model, detailing the baselines, datasets, and results. Lastly, Section 4.5 discusses how the results of the study answer the research questions.

4.2 Implementation Settings

The implementation of both proposed models, LSTM-DynAE and ConvLSTM-DynAE, was carried out using the same experimental configuration. This section presents the software environment, hardware specifications, hyperparameters, and training configurations adopted for all experiments. These settings are described here to ensure the reproducibility of results presented in the following sections.

4.2.1 Implementation Settings of LSTM-DynAE

The experiments in this work are implemented on Google Colab. Google Colab is an online Jupyter Notebooks environment from Google that allows anyone to write and share Jupyter Notebooks in Python online from anywhere. As this work is focused on deep-embedded clustering using an autoencoder-based model, the architecture of the model used is similar to other methods. However, the used autoencoder is an LSTM-AE which is adopted to be suitable for capturing discriminative features from sequential datasets. Moreover, the model has been built using a single LSTM layer in the encoder and the decoder parts of the autoencoder, and it is a fully connected network. For the pretraining phase, the LSTM-AE is trained adversarially end-to-end competing with a critic network for 13×10^4 iterations. The parameters of the autoencoder are updated by Adam optimization function (Kingma & Welling, 2013) with 0.0001 learning rate, and default values for α_1 and α_2 , while τ_1 and τ_2 are set according to (Berthelot, Raffel, Roy, & Goodfellow, 2018b). For the clustering phase, the autoencoder is trained until it meets the convergence criteria with a threshold of 1%, or when it reaches the maximum number of iterations. The SGD of a learning rate $\vartheta=0.001$ and 0.9 momentum value is used for updating the training parameters. Moreover, the pretraining and the clustering phases have been executed in batches of 256 for each.

4.2.2 Implementation Settings of ConvLSTM-DynAE

All implementations of the deep learning algorithms were conducted on the Google Colab-atory Pro platform and Jupyter Lab, where both offer high RAM and GPU capabilities, while the preprocessing was performed on a personal computer with Intel® Core™ i7-14700K processor, 32 GB virtual memory, and a GPU. The steps of the experiments start with data preprocessing, then clustering, and obtaining the load profiles of the generated clusters.

For the ConvLSTM-DynAE, the encoder consisted of two ConvLSTM2D layers with 16 and 32 units, kernel size (1, 7), and ‘same’ padding, each followed by batch normalization and dropout (0.2). The decoder was symmetrically designed with 32 and 16 ConvLSTM2D layers using the same kernel size, and a final TimeDistributed dense layer with ReLU activation to reconstruct the input sequence.

4.2.2.1 Data Preprocessing

Given the nature of the smart meter readings, this experiment employed big-data techniques for loading and manipulating the dataset. The data was read and aggregated from 168 files using `dask` method. `Dask` (Team, 2023), is an open-source parallelization platform that supports Pandas dataframes and NumPy arrays, was utilized due to its high efficiency in handling large datasets. `Dask` dataframes scale Pandas workflows, enabling programs to manage substantial amounts of data by processing it in parallel on a personal laptop or a cluster of computers.

The dataframe was structured so that the house number served as the primary key, allowing for the clustering of houses with similar patterns into the same category, as detailed in sub-Section 3.1. Additionally, 26 houses with less than 20% of readings throughout the analysis were excluded. Subsequently, the missing remaining null values were filled according to Equation 3.31. Moreover, the first 5 days and the last 5 days of the dataset were excluded to shape the dataset into weekly sequences to make it suitable for the size of the filter used in ConvLSTM layers, which has a kernel size of (1,7). The preprocessing resulted in the creation of the (x_i) matrix with dimensions (5540, 39312) for London dataset. That means we have 819 days, which can be divided into 117 weeks, where the week starts from Monday according to UK. Following this, all readings were normalized using Equation 3.32. After normalization, the data was reshaped to be suitable for processing by ConvLSTM autoencoder in a timeseries shape. For other based models, the data shape is kept as two dimensions. The same preprocessing has been done with Irish dataset resulting with a matrix of shape (6345,25776).

4.2.2.2 Clustering

To evaluate our clustering algorithm for load profiling, the dataset was input into the ConvLSTM-DynAE deep clustering networks. The Silhouette Index (Sil) was calculated for values of k ranging from 2 to 10 to identify the optimal number of clusters. The elbow and knee methods were then employed to ascertain the best value of k , as outlined in steps 6-10 of Algorithm 3.

The architecture of our model consists of 13 layers. The encoder consists of 7 layers, and the decoder has 6 layers comprising the reshape layer used to reshape the latent space. For other baseline methods, all network architectures were adapted from the original studies (Eskandarnia et al., 2022; Mrabah et al., 2019; Xie et al., 2016; Guo et al., 2017a; W. Yang et al., 2022). The experimental results indicated that the architecture of two ConvLSTM2D

with 16, 32 units in the encoder and two ConvLSTM2D layers in the decoder with 32, and 16 units, performed best for both datasets. ConvLSTM2D layers used *tanh* activation function, where the final layer utilized the ReLU activation function.

During the pretraining phase, the autoencoder was trained adversarially in an end-to-end manner with a critic network for 1000 iterations *MaxItr*. For this phase, all the learnable parameters ϕ_e , ϕ_d , and ϕ_C , are updated using Adam with a learning rate of 0.0001, and default values of α_1 and α_2 . The hyperparameters τ_1 and τ_2 are set as the original study performed by (Berthelot et al., 2018a).

After pretraining, the autoencoder weights were fine-tuned by optimizing a dynamic loss function. K-means clustering was used to initialize and update the embedded clustering centers. As described in Equation 3.26, the loss function consists of two components: the reconstruction cost L_1 , defined in Equation 3.21, and the clustering loss L_2 , defined in Equation 3.24. The loss function was optimized using Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and a momentum of 0.9. Both the pretraining and clustering phases were executed in batches of 256 samples, with the optimization process running for varying numbers of iterations.

For the learning rate of 0.0001, the autoencoder weights were saved. To ensure a fair comparison, the weights saved from the pre-training stage were loaded into the clustering layer of the clustering networks. In the clustering phase, the model was trained for 1000 epochs until it meet the convergence criteria with a convergence threshold *total* = 1% or reaching the maximum number of iterations *MaxItr*. The confidence threshold (κ) is set to $((\kappa)=0.3 \times K)$ and its dropping rate is equal to 0.3 of the threshold (κ). The experimental results indicated that the number of iterations for the clustering phase should be determined when the network reached convergence, which for the experimental datasets occurred around 100 iterations.

4.2.2.3 Evaluation and Generating Load Profiles

Based on the clustering results, various assessment criteria can be employed to evaluate the performance of load profiling algorithms, which will assist in determining an appropriate number of clusters. As outlined in Algorithm 3, the proposed framework utilizes the Silhouette Index (*Sil*) for this evaluation. Additionally, the objective of the load profiling task is to identify distinct groups of consumers. This goal is achieved when the resulting High Energy Consumption (HEC) patterns exhibit variability in both consumption scales and intra-day fluctuations (Sun et al., 2020). The quality of the load curves can also be

visually assessed by examining the load curve for each group, as detailed in steps 24–27 of Algorithm 3.

To derive the load profile for each cluster, the centroids of each cluster are calculated. Using the Python datetime package, three empty time series of size 819 were created, with timestamps from 00:00 AM on November 28, 2011, to February 23, 2014. The average consumption of all houses within each cluster at each timestamp is then populated into these time series, as described in steps 17–27 of Algorithm 3. These time series were subsequently re-sampled to a daily frequency, and the normal load profile for each cluster was visually presented as shown in Figure 4.2 and Figure 4.1

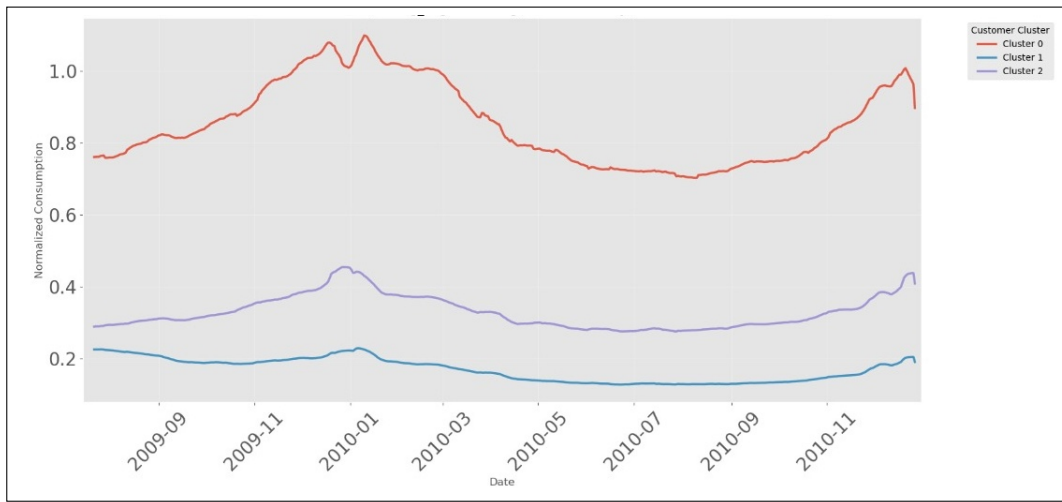


Figure 4.1: Load Profiles of Irish Dataset obtained by ConvLSTM-DynAE

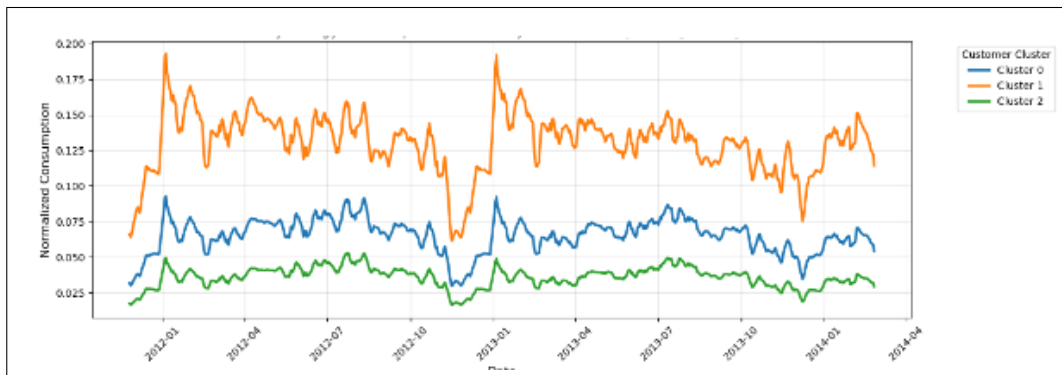


Figure 4.2: Load Profiles of London Dataset obtained by ConvLSTM-DynAE

4.3 Results of LSTM-DynAE Model

In this section, the LSTM-DynAE model is examined on several benchmark datasets to evaluate the performance of the LSTM-DynAE model on sequential datasets. In the following subsections, the baselines and datasets used, and the results are presented.

4.3.1 Baselines and Datasets

4.3.1.1 Baselines

The selection of the baselines depends on two main factors. The first factor is comparing both the original DynAE with the proposed model LSTM-DynAE model to show the difference between using the DynAE and the LSTM-DynAE with sequential data. The second factor is comparing the proposed LSTM-DynAE model with other state-of-the-art models used for clustering sequential data. This factor must demonstrate that the enhanced model is able to compete with existing models utilizing LSTM technology. Additionally, a new model for Deep Embedded Clustering in sequential datasets was proposed, utilizing a dynamic loss function and an LSTM-AE architecture, which should be evaluated against other innovative models. According to the aforementioned information, the DynAE (Mrabah, Khan, et al., 2020), DEC (Xie et al., 2015), ADEC (Mrabah, Khan, et al., 2020), and IDEC (Guo, Gao, Liu, & Yin, 2017b) have been chosen to address the initial factor. The DTC (Madiraju, 2018), IT-TSC (Xu, Huang, & Yoo, 2021b), and DTCR (MacQueen, 1967) are chosen to confront the second factor.

4.3.1.2 Datasets

In this study, the experiments concentrate on five time series datasets with fixed-length time steps, which have been selected from the UEA Multivariate Time Series Classification archive. All datasets can be accessed at <https://timeseriesclassification.com>. Table 4.1 outlines the characteristics of the datasets utilized in this research. As the clustering process is unsupervised, the training and testing sizes are combined for clustering purposes. The number of classes corresponds to the number of labels present in each dataset. Since the models are designed to learn an efficient, low-dimensional representation of the entire dataset for clustering, the full dataset was used for the unsupervised training of the Auto-Encoder and the subsequent clustering.

Table 4.1: Characteristics of the Selected Time Series Datasets

Code	Dataset	Train Size	Test Size	Dimensions	Length	Classes
PD	PenDigits	7494	3498	2	8	10
HW	Handwriting	150	850	3	152	26
HB	Heartbeat	204	205	61	405	2
RS	RacketSports	151	152	6	30	4
FD	FaceDetection	5890	3524	144	62	2

4.3.2 Results

The results of the experiments are presented and discussed in this section. They are divided into three parts: comparison with state-of-the-art methods, effects of the dynamic loss function, and Feature Drift and Feature Randomness.

4.3.2.1 Comparing Results with the State-of-the-Art Methods

This section shows the results of the clustering performance of the LSTM-DynAE method proposed in this work in comparison with the state-of-the-art methods. Five sequential datasets are utilized to compare the performance of the clustering methods. Table 4.2 depicts the comparison results of the clustering performance of LSTM-DynAE, DynAE, DTC, IT-TSC, DEC, ADEC, and IDEC methods in terms of *Acc* and *NMI* metrics. As shown in Table 4.2, the LSTM-DynAE method outperforms the comparable methods in *Acc* and *NMI*. The highest clustering performance in LSTM-DynAE method is recorded by the Heartbeat dataset, with a value of 0.97, and 0.95, for *Acc* and *NMI*, respectively. Meanwhile, the lowest performance is recorded by the Handwriting dataset, with a value of 0.78, and 0.74, for *Acc* and *NMI*, respectively. However, both datasets record the best results with the LSTM-DynAE method in comparison with other state-of-the-art methods. It can be observed from the results, that the DynAE, DTC, IT-TSC, and DEC results have the highest *Acc* and *NMI* performance with RacketSports, and that can be related to the pre-processing stage done with the data. However, the LSTM-DynAE method still outcomes the best performance with the same dataset, as the *Acc* and *NMI* are 0.92, and 0.90 for *Acc* and *NMI*. Another observation that can be recorded from the results is that the Heartbeat dataset records the best results in clustering with LSTM-DynAE, IT-SC, DTCR, ADEC, and IDEC methods. The results also illustrate that the proposed LSTM-DynAE method is effective in clustering sequential datasets with dynamic learning capability. Moreover, the

DynAE is not suitable to be used with these datasets.

Table 4.2: Comparison of the Clustering Performance of LSTM-DynAE with State-of-the-art Clustering Methods in terms of ACC AND NMI Metrics - (Best results in bold, second best emphasized)

Dataset	<i>LSTM-DynAE</i>		DynAE		DTC		IT-TSC		DTCR		DEC		ADEC		IDEC	
	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>	<i>Acc</i>	<i>NMI</i>
PenDigits	0.82	0.88	0.68	0.62	0.79	0.71	0.64	0.60	0.71	0.69	0.54	0.52	0.64	0.60	0.61	0.59
Handwriting	0.78	0.74	0.58	0.53	0.68	0.66	0.57	0.50	0.67	0.67	0.44	0.43	0.57	0.50	0.48	0.49
Heartbeat	0.97	0.95	0.66	0.63	0.71	0.70	0.68	0.69	0.69	0.69	0.58	0.53	0.68	0.69	0.63	0.62
RacketSports	0.92	0.90	0.75	0.75	0.82	0.79	0.67	0.64	0.61	0.62	0.65	0.64	0.67	0.64	0.62	0.62
FaceDetection	0.84	0.81	0.55	0.54	0.60	0.61	0.75	0.75	0.65	0.63	0.42	0.43	0.55	0.52	0.54	0.55

To evaluate the performance of the LSTM-DynAE method, the execution time of the proposed method is compared with the other clustering methods, DynAE, DTC, IT-TSC, DTCR, DEC, ADEC, and IDEC. Table 4.3 shows the execution time of the LSTM-DynAE and the state-of-the-art methods with the five sequential selected datasets. It can be observed that the execution time of DEC and ADEC are higher than other clustering methods. Moreover, the sequential methods, DTC, IT-TSC, and DTCR have lower execution time than other non-sequential methods, DEC, ADEC, IDEC, and the DynAE methods. On the other hand, the execution time of the proposed LSTM-DynAE method is lower than the sequential methods, DTC, IT-TSC, and DTCR. Besides, the execution time of the LSTM-DynAE method is lower than DynAE when applied to the state-of-the-art datasets. Consequently, it can be concluded that the proposed LSTM-DynAE method surpasses other methods in execution time when applied to sequential datasets.

Table 4.3: Execution Time of LSTM-DynAE and Baseline Models on Benchmark Datasets (in ms)

Dataset	LSTM-DynAE	DynAE	DTC	IT-TSC	DTCR	DEC	ADEC	IDEC
PenDigits	4320	9225	4587	4487	4474	10886	9445	9335
Handwriting	5085	9160	6087	5087	5560	9788	9756	9835
Heartbeat	5040	9665	10554	8554	7522	15230	10664	10246
RacketSports	2880	9442	5885	4885	4325	10998	10550	10446
FaceDetection	2500	7662	4728	4660	4371	4992	4776	4880

4.3.2.2 Effect of Dynamic Loss Function

This section discusses how the dynamic loss function affects the model to train smoothly and gradually using the smooth dynamic loss function. As the Heartbeat dataset has the best performance when the LSTM-DynAE model is applied, the dynamic learning during the training of this dataset has been selected to show the effect of the dynamic loss function on the training process. Figure 4.3 shows the clustering loss of the dynamic loss function during the training of the Heartbeat dataset. It is clear from the figure that the clustering loss decreases smoothly and gradually. That means the smooth and gradual change in the dynamic loss function is suitable for the learning process. Moreover, as shown in Figure 4.4, the values of the *Acc* and *NMI* during the training process of Heartbeat dataset, it has been observed that both metrics are increasing smoothly and gradually. Consequently, this smooth increase in both metrics during training depicts the smooth dynamic learning via

the dynamic loss function utilized for sequential data clustering.

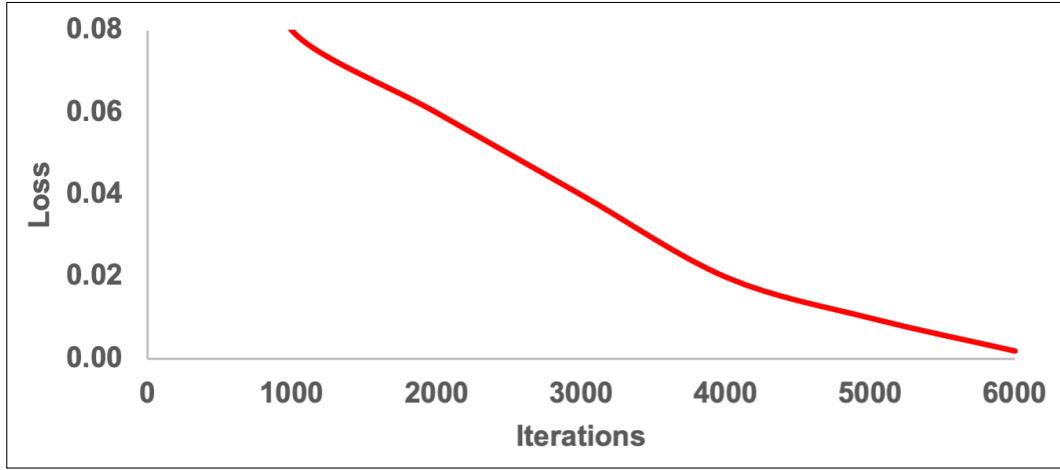


Figure 4.3: Clustering Loss of Dynamic Loss Function During Training on Heartbeat Dataset

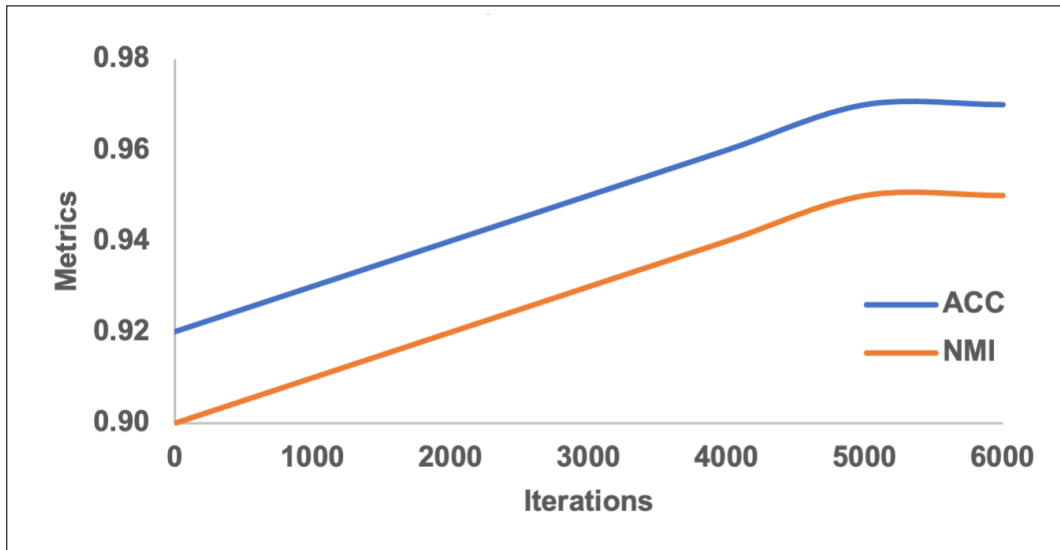


Figure 4.4: Acc and NMI of LSTM-DynAE During the Training of Heartbeat Dataset

4.3.2.3 Feature-Randomness and Feature-Drift

LSTM-DynAE utilized the dynamic loss objective function to solve the trade-off between FR and FD.

The provided graphs illustrate the ΔFR during the training of the Heartbeat dataset using two different models, DTC and LSTM-DynAE. In the left graph, focusing on the

performance of the DTC, the ΔFR fluctuates significantly throughout the iterations. It starts at a moderate level, dips at various points, and then stabilizes towards the end of the training process. The average ΔFR , remains relatively constant at a lower value, suggesting that while the FR experienced fluctuations, it did not exceed the average threshold noticeably. Conversely, the right graph showcases the ΔFR under the LSTM-DynAE model, which presents a markedly different trend. The ΔFR starts low but dramatically increases after the initial iterations, eventually attaining a plateau at a higher value. The average feature randomness remains constant as well, but in this case, the actual ΔFR remains consistently above the average (0.97), indicating a higher stability and performance during training compared to the DTC model.

In summary, the comparison of ΔFR in the two models reveals distinct behaviors. The DTC exhibits more volatility and generally lower performance, while the LSTM-DynAE shows a rising trend and higher levels of FR, indicating its effectiveness in modeling and learning from the timeseries dataset.

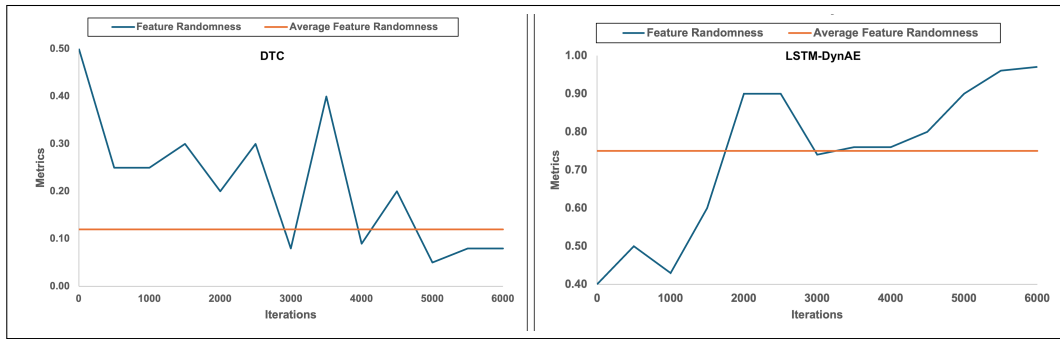


Figure 4.5: Comparison of ΔFR between DTC and LSTM-DynAE During the Training of Heartbeat Dataset

Figure 4.6 illustrates the concept of FD during the training of Heartbeat dataset using DTC model (on the left) and LSTM-DynAE (on the right). Each graph shows how the metrics of FD evolve over 6,000 training iterations. In the first graph, which focuses on the DTC method, ΔFD fluctuates significantly throughout the iterations, demonstrating a visible variation in metrics. The blue line represents the ΔFD , while the orange line indicates the average FD across the iterations. Notably, the ΔFD reaches peaks and troughs, indicating that the performance of the model may be unstable during certain phases of training. In contrast, the second graph that shows the LSTM-DynAE approach depicts a different trend. The ΔFD maintains a more stable trajectory with fewer oscillations compared to the DTC graph. The blue line indicates that while there is still some fluctuation, it averages

out to a higher level over time, indicating that the LSTM-DynAE model is better suited for managing FD effectively.

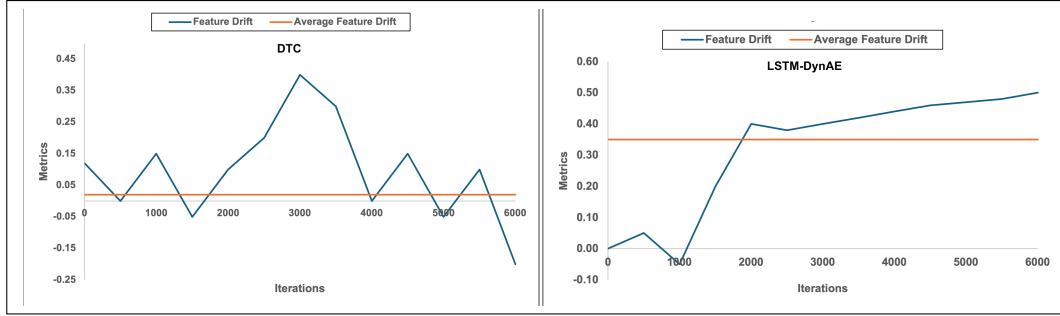


Figure 4.6: Comparison of ΔFD between DTC and LSTM-DynAE During the Training of Heartbeat Dataset

The sharp drop in the performance of DTC model around iteration 3000, in contrast to the DynAE drop, fundamentally represents an aggressive reorganization versus a defensive stabilization. The DynAE drop is an instability event caused by FD and the immediate recovery confirms that its specialized dynamic loss successfully acted as a stabilizing regulator to pull features back into a preserved, cohesive manifold. Conversely, the DTC drop is not a failure of stability but a side effect of the sudden, forceful activation of the external contrastive loss objective, which is designed to deliberately shatter the existing feature structure to maximize the distance between different time series samples. This brief, violent re-organization temporarily disrupts the reconstruction objective, causing the metric to drop; however, the subsequent rapid recovery validates that this aggressive process achieves a superior, highly discriminative latent space where the samples are more optimally separated for clustering than was possible under a stability constraint alone.

4.4 Results of ConvLSTM-DynAE Model

To demonstrate the effectiveness of ConvLSTM-DynAE model in improving the load profiling process, we conducted extensive experiments. This section details the application of Algorithm 3 combined with ConvLSTM-DynAE Algorithm 2 on real world datasets to evaluate its performance. We begin by describing the baselines and datasets used, followed by a presentation and analysis of the results.

4.4.1 Baselines and Datasets

4.4.1.1 Baselines

The analysis of Silhouette Scores for different cluster counts was performed for the ConvLSTM-DynAE algorithm, along with five deep learning methods: DEC, IDEC, DynAE, and LSTM-AE. To evaluate the proposed enhanced load profiling method, these deep learning clustering algorithms were compared to several traditional clustering techniques, including K-Means, hierarchical clustering, GMM, and spectral clustering. The conventional methods were executed with various parameters, recording the highest Silhouette Score (*Sil*) for each. In contrast, the original parameters from the initial studies were maintained for all other deep clustering methods.

4.4.1.2 Datasets

In this study, two open smart-meter datasets are utilized to evaluate our proposed method. The datasets originate from the UK Power Networks in London (United Kingdom)(UK Power Networks, 2014), and the smart metering trial conducted collaboratively by the Commission for Energy Regulation (CER) in Ireland (Commission for Energy Regulation (CER), 2012). This initiative involved meticulously recording the electricity consumption of customers at thirty-minute intervals, resulting in a comprehensive and valuable source of data for our analysis. The datasets are described in the following lines:

- **Low Carbon London Smart Meter dataset:**

This dataset contains energy consumption data from a sample of 5,567 households in London that participated in the Low Carbon London project, organized by UK Power Networks. The data was collected from November 2011 to February 2014, with readings taken every half hour. To ensure the sample is representative, participants were selected to reflect the demographics of the Greater London population. The dataset provides information on energy consumption in kilowatt-hours (kWh) for each half-hour period, along with unique household identifiers and corresponding date and time stamps. When uncompressed, the CSV file occupies about 10 GB of storage and include approximately 167 million rows. The London dataset consists of five main columns:

- LCLid: A unique identifier assigned to each household record.

- stdorToU: This field indicates whether the consumer is on a Dynamic Time of Use (dToU) tariff.
- DateTime: Time stamps recorded at half-hourly intervals.
- KWH/hh (per half hour): This column captures the energy consumption in kilowatt-hours (kWh) for each household.

- **Irish Dataset:**

The Commission for Energy Regulation (CER) acts as the regulatory authority for Ireland’s electricity and natural gas sectors. In 2007, the CER launched the Smart Metering Project to conduct trials and evaluate the effectiveness of Smart Meters. Irish data is organized into six zipped files, when unzipping the files, it was revealed that there are 6,445 houses with meter IDs ranging from 1000 to 7444. Each zipped file contains a single text file includes three columns:

- Meter ID: A unique identifier for the smart meter
- Five-digit code: The first three digits represent the day code (where day 1 corresponds to January 1, 2009). The last two digits indicate the time code, which ranges from 01 to 48 for each 30-minute interval (for example, 1 signifies the time period from 00:00:00 to 00:29:59).
- Electricity consumed during a 30-minute interval (in kWh): Contains energy consumption in kilowatt per half hour for each smart meter.

The ConvLSTM-DynAE model was specifically designed for load profiling tasks, which involve spatiotemporal data (data with both spatial and temporal correlations, like smart meter readings from multiple households). In contrast, the earlier LSTM-DynAE model was validated on benchmark sequential datasets (e.g., Heartbeat, RacketSport, etc.) that capture only temporal dependencies, not spatial structures. So, reusing the same datasets would not demonstrate the advantage of ConvLSTM layers, which are meant to learn spatial–temporal relationships.

4.4.2 Results

As outlined in Algorithm 3, step 13, the elbow method was employed to determine the optimal number of clusters. The results indicate a clear break at $k=3$ for ConvLSTM-DynAE

method for London and Irish datasets as shown in Figure 4.7 and Figure 4.8. The experimental findings reveal that, for all clustering algorithms used in load profiling validation and implementation, $k=3$ was the most suitable number of clusters.

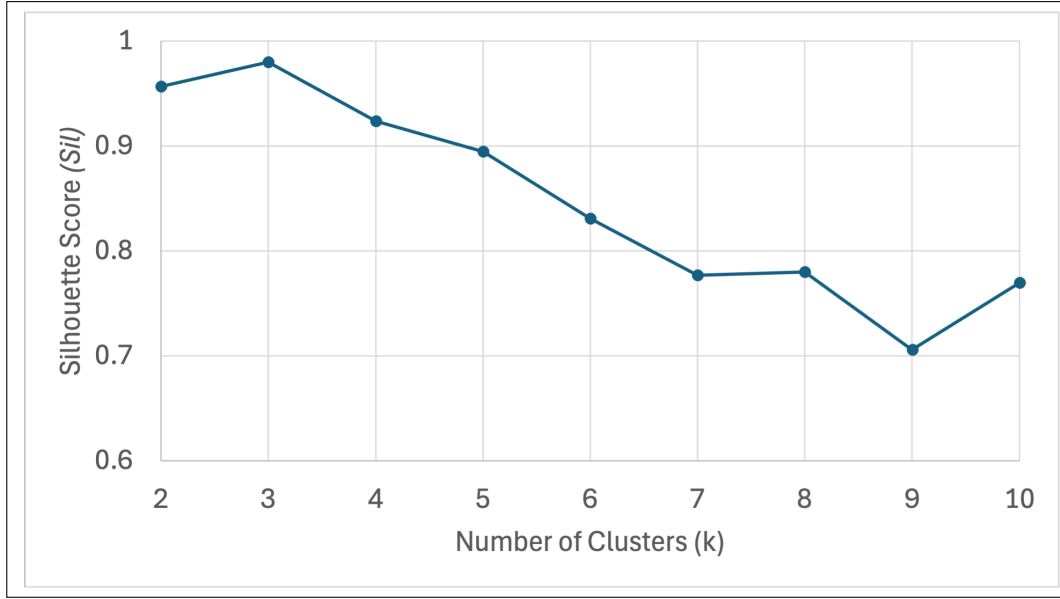


Figure 4.7: Elbow methods on Silhouette Score (Sil) for ConvLSTM-DynAE clustering using London dataset

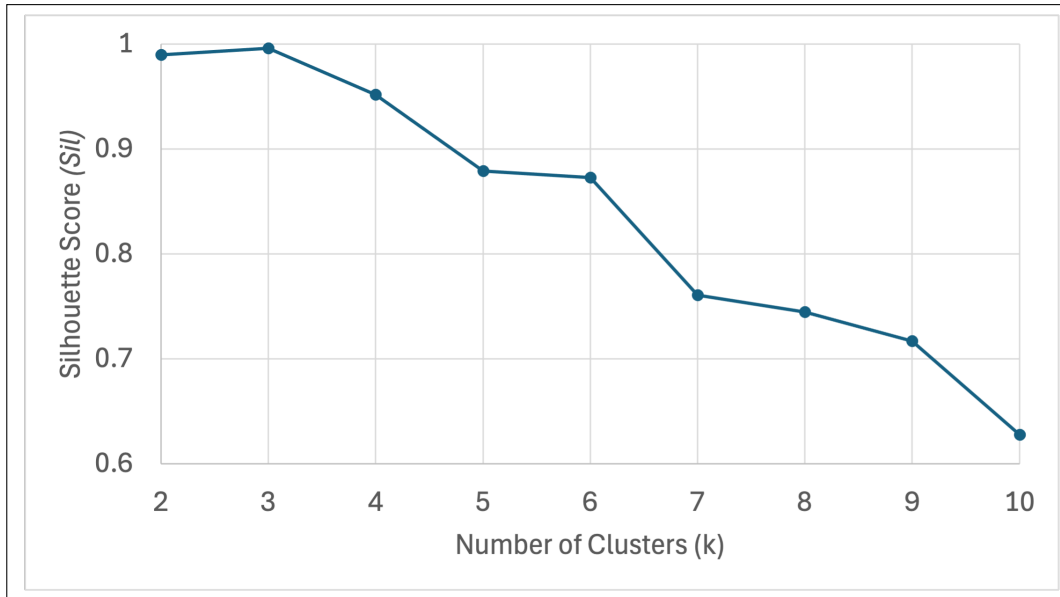


Figure 4.8: Elbow methods on Silhouette Score (Sil) for ConvLSTM-DynAE clustering using Irish dataset

Table 4.4 and Table 4.5 indicate that deep clustering markedly enhanced the *Sil* score in London dataset, and Irish dataset, respectively. *Sil* score recorded 0.824 for DynAE and 0.980 for ConvLSTM-DynAE in London dataset. For Irish, the value of *Sil* obtained by DynAE was 0.780, and 0.996 obtained by ConvLSTM-DynAE. In both datasets, *Sil* obtained by our proposed method ConvLSTM-DynAE achieved the best performance, showcasing the efficiency of the proposed framework for load profiling. It is important to note that our model effectively balances feature randomness and feature drift due to the integration of the dynamic loss function used in DynAE, as discussed by (Mrabah, Khan, et al., 2020) in Section 3.1 and 3.2. The experimental outcomes, based on a real-world datasets, suggest that ConvLSTM-DynAE is particularly well-suited for load profiling tasks.

Table 4.4: Silhouette Index (*Sil*) results for conventional vs deep clustering methods for the London Dataset

No. of Clusters	Conventional Clustering				Deep Clustering				
	K-Means	Hierarchical	GMM	Spectral	DEC	IDEC	DynAE	LSTM-AE	ConvLSTM-DynAE
2	0.441	0.487	0.180	0.437	0.728	0.742	0.739	0.650	0.957
3	0.601	0.566	0.332	0.601	0.813	0.816	0.824	0.773	0.980
4	0.341	0.227	0.138	0.343	0.649	0.665	0.712	0.528	0.924
5	0.304	0.229	0.108	0.285	0.626	0.612	0.728	0.643	0.895
6	0.306	0.231	0.097	0.236	0.613	0.616	0.690	0.570	0.831
7	0.216	0.192	0.091	0.191	0.608	0.618	0.631	0.542	0.777
8	0.211	0.193	0.067	0.191	0.542	0.550	0.621	0.520	0.780
9	0.194	0.198	0.069	0.184	0.563	0.573	0.612	0.489	0.706
10	0.217	0.198	0.059	0.145	0.485	0.495	0.545	0.475	0.770

Table 4.5: Silhouette Index (*Sil*) results for conventional vs deep clustering methods for the Irish Dataset

No. of Clusters	Conventional Clustering				Deep Clustering				
	K-Means	Hierarchical	GMM	Spectral	DEC	IDEC	DynAE	LSTM-AE	ConvLSTM-DynAE
2	0.454	0.371	0.022	0.327	0.613	0.622	0.742	0.677	0.990
3	0.614	0.645	0.341	0.441	0.723	0.751	0.801	0.780	0.996
4	0.442	0.375	0.051	0.302	0.558	0.561	0.710	0.660	0.952
5	0.363	0.328	0.035	0.242	0.571	0.612	0.730	0.681	0.879
6	0.379	0.325	0.030	0.284	0.580	0.623	0.689	0.652	0.873
7	0.293	0.297	0.050	0.216	0.584	0.642	0.624	0.614	0.761
8	0.311	0.297	0.051	0.187	0.584	0.570	0.624	0.549	0.745
9	0.241	0.192	0.051	0.185	0.562	0.573	0.610	0.558	0.717
10	0.240	0.194	0.045	0.182	0.561	0.580	0.565	0.532	0.628

The results indicate that the *Sil* score improves through the training iterations as shown in Figure 4.9, this could compromise the quality of the load profiles. As highlighted by

(Panapakidis & Christoforidis, 2017) and (Zhang, Huang, & Bompard, 2018), the load profiling task encompasses not just clustering, but also the choice of pattern representation techniques and the extraction of representative consumer profiles. The goal is to effectively differentiate consumer groups and generate a meaningful load curve for each cluster. Consequently, as stated in Algorithm 1, the resultant *Sil* value, along with visualizations of HEC, is utilized to select the optimal algorithm for load profiling. Figures 4.1 and 4.2 depict the load profiles generated by ConvLSTM-DynAE for Irish and London datasets, respectively. The figures visually show distinctly separable load curves for each cluster.

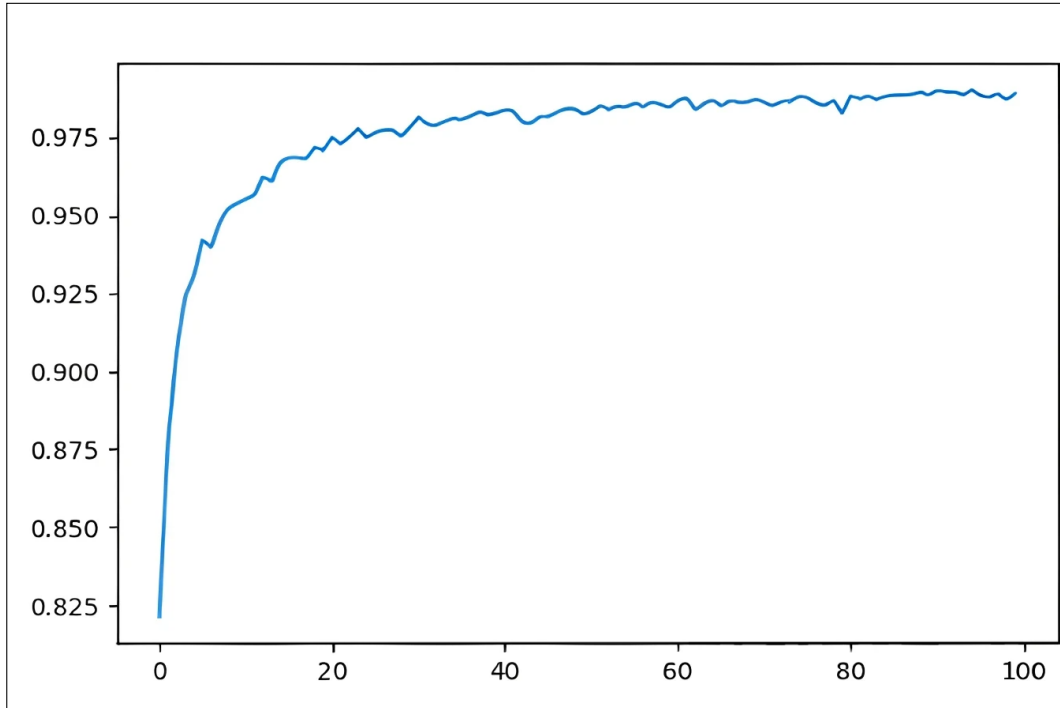


Figure 4.9: *Sil* Score per Iteration for London Dataset

One notable observation that DynAE outperforms the LSTM-AE on the large, complex load profiling datasets (London and Irish) because its architecture introduces a critical refinement mechanism designed to counteract the instability issues of deep clustering. While the standard LSTM-AE relies primarily on reconstruction loss, allowing the features in the latent space to experience FD and FR during the joint clustering process, DynAE incorporates a dynamic regularization loss that forces the latent space to be stable and well-separated for clustering. This dynamic mechanism serves to continuously refine the latent embeddings based on the evolving cluster assignments, ensuring the final representation is robust and optimized for cluster purity rather than just data fidelity. This superior stability and latent space refinement prove essential for accurately clustering the noisy, long-term

consumption patterns found in the London and Irish datasets, whereas this advantage was not as apparent or necessary in the previous, simpler benchmark datasets.

To validate that deep clustering outperforms classical methods and to provide a benchmark result, the framework results obtained by the Irish dataset are compared with the results obtained by the same dataset in (Eskandarnia et al., 2022), (Kumar & Mallipeddi, 2024), and (Sun et al., 2020). For London dataset, the results are compared with the first two studies (Eskandarnia et al., 2022), and (Kumar & Mallipeddi, 2024) as they used the same dataset. In (Sun et al., 2020), the authors employed a concurrent k-means algorithm for dimensionality reduction followed by spectral clustering for obtaining clusters. While the works conducted by (Eskandarnia et al., 2022) and (Kumar & Mallipeddi, 2024) employed auto-encoder based deep embedded clustering methods in which dimension reduction is performed by the autoencoders similar to our method. Since DynAE achieved the best performance in (Eskandarnia et al., 2022), we compare our results in Irish datasets with DynAE results used in (Eskandarnia et al., 2022), Autoencoder Clustering (AEC) proposed in (Kumar & Mallipeddi, 2024), and CKSC (Sun et al., 2020). The comparison of Silhouette scores (*Sil*) in Table 4.6 and Table 4.7 reveal that our proposed deep clustering method delivered superior results. This validates our hypothesis that the dimensionality reduction and deep features extracted by the autoencoder are more effective than the similarity matrix produced by k-means and spectral clustering reported by (Sun et al., 2020). The charts in Figure 4.10 and Figure 4.11 again confirm that, based on the elbow method, $k=3$ is the optimal number of clusters, demonstrating the enhanced performance of our proposed method which integrates the spatial temporal capabilities of ConvLSTM-AE with dynamic loss function.

Table 4.6: Silhouette Score (*Sil*) for CKSC, DynAE, AEC, and the proposed method (ConvLSTM-DynAE) on the Irish Dataset

No. of Clusters	ConvLSTM-DynAE	DynAE (Eskandarnia et al., 2022)	AEC (Kumar & Mallipeddi, 2024)	CKSC (Sun et al., 2020)
2	0.990	0.740	0.429	0.425
3	0.996	0.616	0.443	0.233
4	0.952	0.529	0.445	0.240
5	0.879	0.509	0.450	0.123
6	0.873	0.517	0.480	0.133
7	0.761	0.540	0.490	0.105

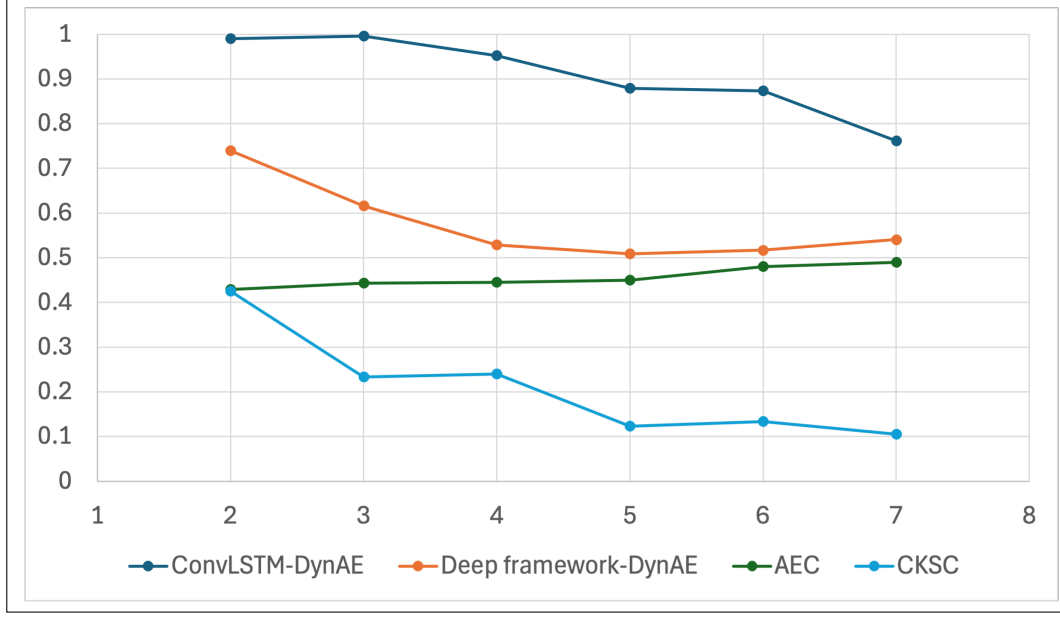


Figure 4.10: Comparison of ConvLSTM-DynAE Results with Previous Works on Irish Dataset

Table 4.7: Silhouette Score (Sil), DynAE, AEC, and the proposed method (ConvLSTM-DynAE) on the London Dataset

No. of Clusters	ConvLSTM-DynAE	DynAE (Eskandarnia et al., 2022)	AEC (Kumar & Mallipeddi, 2024)
2	0.957	0.623	0.580
3	0.980	0.624	0.590
4	0.924	0.634	0.620
5	0.895	0.582	0.620
6	0.831	0.532	0.610
7	0.777	0.437	0.610

4.5 Discussion

After presenting the experimental results, in this section, we intend to answer the research questions. The following subsections answer the questions regarding the two proposed models, LSTM-DynAE and ConvLSTM-DynAE.

RQ1: What are the main challenges associated with clustering high-dimensional time series data?

RQ2: How will deep learning models, particularly autoencoders, be used for dimensionality reduction in time series clustering?

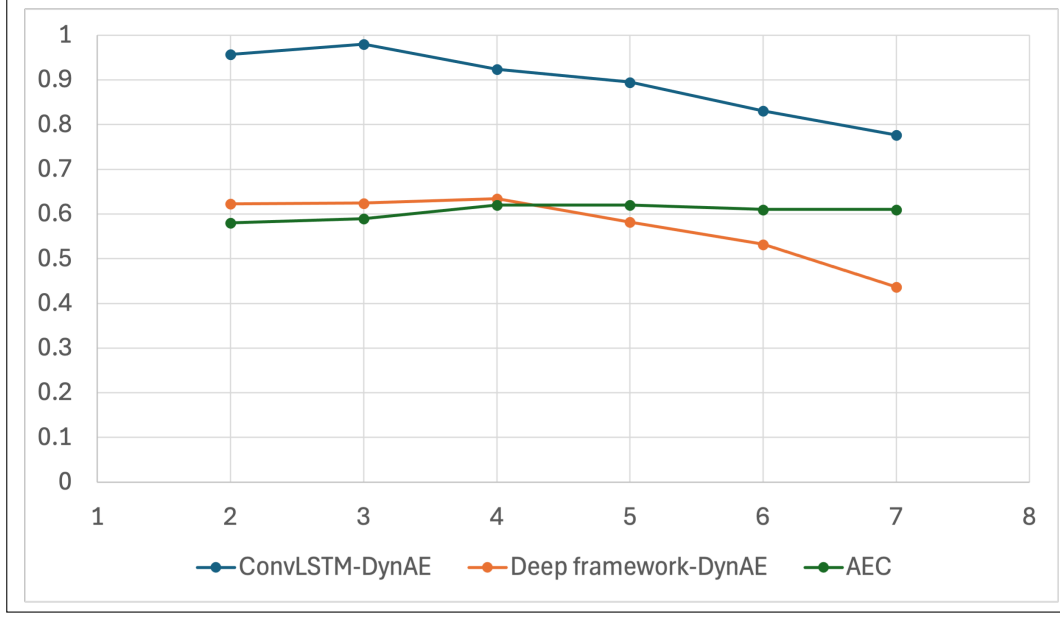


Figure 4.11: Comparison of ConvLSTM-DynAE Results with Previous Works on London Dataset

These two questions are related to each other. The answer of RQ1 is illustrated in (Chapter-1/Section 1.2), and in the Literature Review (Chapter-2/ Section 2.2.1). While the answer of RQ2 is depicted in (Chapter 2/ Section 2.2.4).

Regarding RQ1, it is demonstrated from the both sections that clustering high-dimensional time series data faces several significant challenges. One of the primary issues is the curse of dimensionality, which leads to sparsity in the feature space as the number of dimensions increases. This sparsity makes it difficult to identify meaningful patterns and relationships within the data, as traditional distance metrics become less reliable. Additionally, time-series data are characterized by strong temporal dependencies, which must be explicitly modeled to accurately capture the dynamics of the data. Traditional clustering methods often fail to account for these dependencies, resulting in suboptimal clustering outcomes. Furthermore, the presence of noise and missing values in time series data complicates the clustering process, as it can distort the underlying patterns. Lastly, the lack of labeled data restricts the use of supervised learning methods, making unsupervised learning techniques, particularly clustering, essential for exploring the structures within the data. The answer of this question is aligned to research objective 1(RO1: To conduct a comprehensive literature review to identify and categorize the key challenges faced in clustering high-dimensional time series data).

For RQ2, the literature showed that deep learning models, especially autoencoders,

have emerged as powerful tools for dimensionality reduction in time series clustering. Autoencoders learn efficient representations of data through unsupervised learning by compressing the input data into a lower-dimensional latent space and then reconstructing it. This process allows autoencoders to capture complex patterns and temporal dependencies inherent in time series data. Specifically, LSTM-AE are particularly effective for sequential data, as they can model long-term dependencies and handle irregular time intervals. The integration of autoencoders with clustering objectives has led to the development of deep embedded clustering methods, which optimize both feature learning and clustering assignments simultaneously.

The results from the proposed LSTM-DynAE and ConvLSTM-DynAE models provide strong evidence that they effectively address these challenges as shown in Chapter 4. For instance, the LSTM-DynAE model achieved an impressive *Acc* of 0.97 and an *NMI* of 0.95 on the Heartbeat dataset, it also achieved an *Acc* of 0.92 and an *NMI* of 0.90 on the RacketSports dataset, as shown in Table 4.2. This high performance indicates that the model is capable of capturing the temporal dependencies and patterns in the data, thereby addressing the challenge of high dimensionality. Similarly, the ConvLSTM-DynAE model demonstrated its effectiveness by achieving a Silhouette Score (*Sil*) of 0.980 on the London dataset and 0.996 on the Irish dataset, as illustrated in Figures 4.5 and 4.6. These results highlighted the ability of the model to manage the complexities of high-dimensional time series data, including noise and temporal dependencies. This approach enhances the quality of the learned representations, making them more suitable for clustering tasks.

Moreover, when compared to other state-of-the-art methods as shown in Table 4.2, the LSTM-DynAE model consistently outperformed models such as DEC, DTC, and IDEC in terms of both *Acc* and *NMI* across various datasets, further supporting its suitability for handling the challenges associated with clustering high-dimensional time series data. The integration of a dynamic loss function in both LSTM-DynAE and ConvLSTM-DynAE also played a crucial role in mitigating the effects of FR and FD, which are significant challenges in clustering high-dimensional data.

Finally, the robustness of the LSTM-DynAE and ConvLSTM-DynAE models in effectively handling noise and missing values in time series data is reflected in their high clustering performance metrics. The architectures of these models are specifically designed to capture temporal dependencies, which is essential for accurately clustering time series data that may be affected by these issues. Overall, the results from the LSTM-DynAE and ConvLSTM-DynAE models provide compelling evidence that they effectively address

the challenges associated with clustering high-dimensional time series data, demonstrating their robustness and capability in managing complex temporal patterns. It showed that deep learning models, particularly autoencoders, are effective for dimensionality reduction in time series clustering, enhancing the quality of learned representations and improving clustering performance. The answer of RQ2 is aligned to the research objective 2 (RO2: To develop and evaluate a novel autoencoder architecture specifically designed for effective dimensionality reduction in high-dimensional time series data).

RQ3: How can recurrent architectures like LSTM and ConvLSTM improve the temporal feature extraction process for clustering sequential data?

The answer of RQ3 is demonstrated in (Chapter 3/ Section 3.1). As illustrated in the section, recurrent architectures such as LSTM and ConvLSTM significantly enhance the temporal feature extraction process for clustering sequential data. LSTMs are specifically designed to capture long-term dependencies through their unique gating mechanisms, which allow them to retain relevant information from previous time steps while discarding irrelevant data. This capability is crucial for time series data, where each observation is influenced by its predecessors. On the other hand, ConvLSTM combines the strengths of convolutional layers with LSTM units, enabling the model to capture both spatial and temporal features simultaneously. This is particularly beneficial for applications involving spatiotemporal data, such as smart meter readings, where understanding both the temporal dynamics and spatial correlations is essential. By leveraging these recurrent architectures, the proposed models can effectively extract meaningful features that improve clustering performance.

The results from the proposed LSTM-DynAE model provide compelling evidence of the effectiveness of LSTM architectures in handling sequential data. The high *Acc* and *NMI* of LSTM-DynAE, as shown in Table 4.2, indicates that the model effectively captures long-term dependencies in the sequential data, which is essential for accurate clustering. Similarly, the ConvLSTM-DynAE model demonstrated its capability by achieving high Silhouette Score (*Sil*) on the London and Irish datasets, as depicted in Figures 4.5 and 4.6. These results highlight the effectiveness of the recurrent architectures to capture temporal features, thereby enhancing the quality of the learned representations for clustering.

Overall, the results from the LSTM-DynAE and ConvLSTM-DynAE models provide strong evidence that recurrent architectures like LSTM and ConvLSTM significantly improve the temporal feature extraction process for clustering sequential data, leading to enhanced clustering performance and more meaningful insights from the data. The answer of

this question is related to the research objective 3(RO3: To investigate the effectiveness of LSTM and ConvLSTM architectures in enhancing the extraction of temporal features for improved clustering performance on sequential data).

RQ4: What are the existing strategies for integrating clustering objectives into representation learning (i.e., Deep Embedded Clustering)?

As established in the literature review (Chapter 2, Section 2.3), strategies for integrating clustering objectives into representation learning fall into two categories: separate and integrated embedded learning. Separate approaches (e.g., DEC) learn features before clustering, often yielding suboptimal results as clustering does not refine the features. Conversely, integrated approaches (e.g., IDEC) simultaneously optimize reconstruction and clustering objectives, dynamically enhancing feature extraction for better discriminative power and clustering outcomes. The proposed LSTM-DynAE and ConvLSTM-DynAE models exemplify this integrated approach via dynamic loss functions. LSTM-DynAE achieved high clustering performance *Acc* and *NMI* (Table 4.2), while ConvLSTM-DynAE scored high (*Sil*) on real world datasets (London and Irish), respectively (Figures 4.5, 4.6). These superior results, outperforming state-of-the-art methods, strongly validate the effectiveness of deep embedded clustering methods by integrating clustering objectives into representation learning. The answer of this question is aligned to the research objective 4(RO4: To analyze and compare existing strategies for integrating clustering objectives into representation learning frameworks, identifying strengths and weaknesses).

RQ5: What is the trade-off between Feature Randomness and Feature Drift in deep clustering, and how has it been addressed in the literature?

The answer of this question is demonstrated in (Chapter 2/ Section 2.4). As discussed in the literature, the trade-off between FR and FD is a significant challenge in deep clustering. To address this, dynamic loss functions are employed to adjust the emphasis on reconstruction and clustering losses during training. Self-supervision and pseudo-supervision play crucial roles in balancing FR and FD. Self-supervision helps learn robust features that mitigate FR, while pseudo-supervision leverages both labeled and unlabeled data to enhance learning. However, pseudo-supervision can introduce noise if pseudo-labels are inaccurate.

The LSTM-DynAE and ConvLSTM-DynAE models demonstrated effective balancing of FR and FD. Figures 4.3 and 4.4 depicted that the LSTM-DynAE model exhibited better stability and performance in managing FR and FD compared to the DTC model. Moreover, the high performance of the two models indicated their ability to balance FR and FD,

showcasing their effectiveness in addressing the trade-off between FR and FD.

Overall, the integration of dynamic loss functions in the LSTM-DynAE and ConvLSTM-DynAE models highlighted their robustness in managing the trade-off between FR and FD, leading to improved clustering performance and stability in feature representations. The answer of RQ5 is aligned to research objective 5 (RO5: To explore the concepts of FR and FD in deep clustering, and propose methods for mitigating their negative impacts based on literature findings).

RQ6: How will load profiling in smart grids benefit from clustering and deep learning techniques?

The answer of this question is provided in (Chapter 2/ Section 2.5), and in (Chapter 5). Load profiling in smart grids has significantly benefited from the application of clustering and deep learning techniques. By utilizing unsupervised clustering methods, utilities can effectively segment consumers based on their electricity usage patterns, even in the absence of labeled data. This capability is crucial for understanding consumer behavior and optimizing energy distribution. Deep learning techniques, particularly those involving autoencoders, enhance the ability to extract meaningful features from high-dimensional time series data generated by smart meters. This results in more accurate and representative load profiles, which are essential for operational planning and resource allocation.

The results of the proposed ConvLSTM-DynAE model have demonstrated strong performance in load profiling tasks. By integrating convolutional layers with LSTM units, this model can capture both spatial and temporal dynamics in energy consumption data. This integration allowed utilities to gain deeper insights into consumer behavior and identify distinct patterns in electricity usage. The capability of the model to effectively manage the complexities of time series data enhanced its utility in real-world applications.

Figures 4.5 and 4.6 clarified how the distinct clusters identified by the ConvLSTM-DynAE model in both the London and Irish datasets. These figures showcased the load profiles of the identified clusters, highlighting the effectiveness of the model in capturing unique patterns in electricity usage. The results indicated that the ConvLSTM-DynAE model not only outperformed several state-of-the-art clustering methods but also provided valuable insights that contribute to the efficiency and sustainability of smart grid operations.

The benefits of the ConvLSTM-DynAE model in load profiling are manifold. It achieved high performance on both the London and Irish datasets, indicating its ability to identify distinct clusters effectively. Additionally, by capturing both spatial and temporal dynamics in energy consumption data, the model enables better optimization of re-

source allocation. Its capability to identify unique patterns in electricity usage also enhances anomaly detection, allowing utilities to spot irregularities in consumer behavior. Furthermore, the clustering capabilities of the model facilitate effective consumer segmentation, enabling utilities to tailor their marketing strategies and energy efficiency initiatives. Overall, the ConvLSTM-DynAE model has proven to be a valuable tool in enhancing load profiling tasks and providing critical insights for smart grid operations. The answer of this question is related to research objective 6(RO6: To assess the impact of clustering and deep learning techniques on the accuracy and efficiency of load profiling in smart grid applications).

RQ7: How does the proposed method perform in both labeled sequential data and real-world smart meter load profiling scenarios?

In (Chapter 4/ Section 4.2), the results of the proposed models answered this research question. LSTM-DynAE and ConvLSTM-DynAE have demonstrated strong performance in both labeled sequential data and real-world smart meter load profiling scenarios. In experiments with benchmark datasets, the LSTM-DynAE model surpassed state-of-the-art clustering methods in terms of *Acc* and *NMI*, indicating its effectiveness in handling sequential data. Similarly, the ConvLSTM-DynAE model excelled in load profiling tasks, achieving high Silhouette Scores (*Sil*) in both the London and Irish datasets. The integration of dynamic loss functions within these models has proven to enhance clustering performance by effectively balancing FR and FD. The results indicated that the two proposed models are not only theoretically sound but also practically applicable, providing valuable insights for utilities in managing energy consumption patterns and improving operational efficiency. The answer of this research question is aligned to research objective 7(RO7: To evaluate the performance of the proposed method using sequentially labeled datasets and real-world smart meter data, focusing on accuracy, NMI, and silhouette score).

4.6 Threats to Validity

This section addresses the potential threats that could undermine the validity of our proposed LSTM-DynAE and ConvLSTM-DynAE models for clustering and load profiling in time-series datasets.

One significant threat pertains to ***data quality issues***. Poor data quality, characterized by noisy, incomplete, or biased data, can adversely affect the performance of the models. If the input data is not reliable, the resulting insights and predictions may also be flawed.

Another concern is *the risk of model overfitting*. Given the complexity of our models, there is a possibility that they may perform exceptionally well on training data but struggle to generalize to new, unseen data. This overfitting can lead to misleading conclusions about the effectiveness of the models in practical applications.

Additionally, the performance of the proposed models is sensitive to *the choice of hyperparameters*. Inadequate tuning of these hyperparameters can result in suboptimal performance, which may skew the results and limit the performance of the model. Furthermore, our *comparison of the proposed models* to state-of-the-art methods may not be comprehensive. There could be other models that outperform our proposals, and failing to consider these alternatives may limit the perceived effectiveness of our work.

The real-world applicability of our models is also a concern. The performance observed in experimental settings may not translate directly to real-world scenarios due to factors such as data availability and computational resources. Finally, *the reliance on specific datasets* may restrict the robustness of our findings. To enhance the robustness of our conclusions, further research is necessary to evaluate the performance of the proposed models across a variety of datasets and contexts.

By recognizing these potential threats, we can implement strategies to mitigate them, thereby ensuring the validity and reliability of our proposed models.

4.7 Summary

This chapter presented the results and discussions related to the proposed LSTM-DynAE and ConvLSTM-DynAE models for clustering and load profiling in time-series datasets. The chapter was structured into three main sections: an overview of the LSTM-DynAE model's outcomes, a detailed examination of the ConvLSTM-DynAE model, and a discussion on how the results addressed the research questions posed in the thesis.

In the first section, the performance of the LSTM-DynAE model was evaluated using several benchmark datasets. The chapter outlined the baselines and datasets used for comparison, as well as the implementation settings. The results indicated that the LSTM-DynAE model outperformed several state-of-the-art methods in terms of accuracy Acc and NMI , particularly excelling on the Heartbeat dataset.

The second section focused on the ConvLSTM-DynAE model, detailing its application in load profiling. Similar to the previous section, it discussed the baselines, datasets, and implementation settings. The results demonstrated that the ConvLSTM-DynAE model

significantly enhanced load profiling performance, achieving high Silhouette Scores (*Sil*) on both the London and Irish datasets.

Finally, the chapter addressed the research questions by analyzing the implications of the results. It highlighted the challenges associated with clustering high-dimensional time series data, the effectiveness of deep learning models for dimensionality reduction, and the advantages of recurrent architectures like LSTM and ConvLSTM in improving temporal feature extraction. The chapter emphasized the robustness of the proposed models in managing FR and FD, ultimately showcasing their practical applicability in real-world scenarios. The conclusion of the chapter presented the main threats that could affect the validity of our proposed models.

Chapter 5

Conclusion and Recommendations

5.1 Overview

This chapter summarizes the key findings of the research conducted on the LSTM-DynAE and ConvLSTM-DynAE models for load profiling and clustering in time-series datasets. It also provides recommendations for future work in this area.

5.1.1 Conclusion

The research successfully developed and validated two novel deep learning models, LSTM-DynAE and ConvLSTM-DynAE, aimed at addressing the challenges associated with clustering high-dimensional time series data. The LSTM-DynAE model demonstrated its effectiveness in capturing temporal dependencies and patterns, achieving high performance values, particularly with the Heartbeat dataset, which recorded an *Acc* of 0.97 and *NMI* of 0.95. These results indicate that the model is capable of accurately identifying clusters within the data, reflecting the ability of the model to learn meaningful representations that preserve the temporal dependencies in time series.

Similarly, the ConvLSTM-DynAE model excelled in load profiling tasks, significantly

improving performance metrics such as the Silhouette Score (*Sil*), achieving a score of 0.980 on the London dataset and 0.996 on the Irish dataset. These high Silhouette Scores suggest that the model effectively distinguished between different consumer load profiles, indicating its robustness in identifying distinct patterns in energy consumption. The ability to achieve such high scores demonstrates the model's capacity to handle the complexities of real-world data, which often includes noise and irregularities.

The integration of dynamic loss functions within both models proved to be a crucial factor in mitigating issues related to FR and FD, which are common challenges in deep clustering. By gradually shifting the focus from reconstruction to clustering during training, the models maintained stable and discriminative feature representations. This approach not only enhanced clustering performance but also improved the interpretability of the learned representations, making them more applicable to real-world scenarios.

Overall, the results indicated that these models not only outperformed several state-of-the-art methods but also provided valuable insights for practical applications in smart grid analytics and energy management. The findings underscore the importance of leveraging advanced deep learning techniques to address the unique challenges posed by time-series data, paving the way for more effective and efficient clustering methodologies in various domains. The successful application of these models in load profiling tasks highlights their potential to contribute significantly to the optimization of energy distribution and consumer segmentation in smart grid environments.

In conclusion, the research has made significant strides in advancing the field of time-series clustering through the development of innovative models that effectively capture temporal dynamics and improve clustering outcomes.

5.2 Recommendations and Future Work

Based on the findings of this research, several recommendations for future work can be made. First, future studies should evaluate the proposed models on a wider variety of datasets to assess their robustness across different contexts and applications. This broader dataset evaluation will help to confirm the models' effectiveness in diverse scenarios.

Second, further research could focus on developing more sophisticated hyperparameter tuning strategies, such as automated optimization techniques, to enhance model performance. Proper tuning of hyperparameters is essential for achieving optimal results and ensuring the models perform well under various conditions.

Additionally, implementing the proposed models in real-world smart grid environments would provide insights into their practical applicability and effectiveness in managing energy consumption patterns. Real-world testing is crucial for understanding how these models can be utilized in operational settings.

Moreover, future work could explore the integration of other advanced architectures, such as attention mechanisms or transformer models, to further improve the performance of clustering in time-series data. These innovations could lead to even more effective models capable of handling complex temporal patterns.

Research should also focus on developing methods to enhance data quality, including techniques for handling missing values and noise in time-series datasets. Improving data quality will significantly enhance the reliability of clustering outcomes and the overall performance of the models.

Finally, the methodologies developed in this research could be applied to other domains, such as healthcare, finance, and environmental monitoring, where time-series data plays a critical role. By addressing these recommendations, future research can build upon the findings of this thesis, further advancing the field of deep learning for time-series analysis and clustering.

References

- Aghabozorgi, S., Shirkhorshidi, A. S., & Wah, T. Y. (2015). Time-series clustering—a decade review. *Information systems*, 53, 16–38.
- Al Khafaf, N., Jalili, M., & Sokolowski, P. (2020). A novel clustering index to find optimal clusters size with application to segmentation of energy consumers. *IEEE transactions on industrial informatics*, 17(1), 346–355.
- Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., & Zhou, J. (2017). Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 65–74).
- Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N., & Ouimet, M. (2003). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16.
- Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining* (pp. 359–370).
- Berthelot, D., Raffel, C., Roy, A., & Goodfellow, I. (2018a). Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint*.
- Berthelot, D., Raffel, C., Roy, A., & Goodfellow, I. (2018b). Understanding and improving interpolation in autoencoders via an adversarial regu-

- larizer. *arXiv preprint arXiv:1807.07543*.
- Brownlee, J. (2017). Long short-term memory networks with python. *Machine learning mastery, 1*, 1–229.
- Cai, D., He, X., & Han, J. (2011, June). Locally consistent concept factorization for document clustering. *IEEE Trans. Knowl. Data Eng.*, 23(6), 902–913.
- Chen, J., Song, W., Zuo, X., Zhao, K., Jin, B., Zhu, D., & Dai, B. (2024). 1dcae-tssamc: Two-stage multi-dimensional spatial features based multi-view deep clustering for time series data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 32(04), 593–623.
- Commission for Energy Regulation (CER). (2012). *Cer smart metering project - electricity customer behaviour trial, 2009–2010* (1st ed.). Irish Social Science, Commission for Energy Regulation (CER). Retrieved from <https://www.ucd.ie/issda/data/commissionforenergyregulationcer>
- Czétány, L., Vámos, V., Horváth, M., Szalay, Z., Mota-Babiloni, A., Deme-Bélafi, Z., & Csoknyai, T. (2021). Development of electricity consumption profiles of residential buildings based on smart meter data clustering. *Energy and Buildings*, 252, 111376.
- Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., ... Keogh, E. (2019). The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6), 1293–1305.
- Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., & Zhang, D. (2015). Yading: Fast clustering of large-scale time series data. *Proceedings of the VLDB Endowment*, 8(5), 473–484.
- Eskandarnia, E., Al-Ammal, H. M., & Ksantini, R. (2022). An embedded deep-clustering-based load profiling framework. *Sustainable Cities*

and Society, 78, 103618.

- Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, pp. 226–231).
- Franceschi, J.-Y., Dieuleveut, A., & Jaggi, M. (2019). Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32.
- Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W., & Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision* (pp. 5736–5745).
- Goncalves, N. M., Giurgiu, I., & Schumann, A. (2020). From time series to euclidean spaces: On spatial transformations for temporal clustering. *arXiv preprint arXiv:2010.05681*.
- Guo, X., Gao, L., Liu, X., & Yin, J. (2017a). Improved deep embedded clustering with local structure preservation. In *Ijcai* (Vol. 17, pp. 1753–1759).
- Guo, X., Gao, L., Liu, X., & Yin, J. (2017b, August). Improved deep embedded clustering with local structure preservation. In *Proceedings of the Twenty-Sixth international joint conference on artificial intelligence*. California: International Joint Conferences on Artificial Intelligence Organization.
- Guo, X., Zhu, E., Liu, X., & Yin, J. (2018). Deep embedded clustering with data augmentation. In *Proceedings of the 10th asian conference on machine learning*. Beijing, China.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Hong, D., Gu, Q., & Whitehouse, K. (2017). High-dimensional time series clustering via cross-predictability. In *Artificial intelligence and*

- statistics* (pp. 642–651).
- Hu, W., Miyato, T., Tokui, S., Matsumoto, E., & Sugiyama, M. (2017). Learning discrete representations via information maximizing self-augmented training. *arXiv preprint*.
- Huang, H., Shah, T., & Yoo, S. (2022). Deep time series sketching and its application on industrial time series clustering. In *2022 IEEE International Conference on Big Data (Big Data)* (pp. 1997–2006).
- Hurst, S. (2010). A generalized asymmetric student-t distribution with application to financial econometrics. *Journal of Econometrics*, 157(2), 297–305.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.
- Jabi, M., Pedersoli, M., Mitiche, A., & Ayed, I. B. (2019). Deep clustering: On the link between discriminative models and k-means. *IEEE transactions on pattern analysis and machine intelligence*, 43(6), 1887–1896.
- Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8), 651–666.
- Julier, S. J., & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3), 401–422.
- Keogh, E., & Kasetty, S. (2002). On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 102–111).
- Khan. (2020). Towards efficient electricity forecasting in residential and commercial buildings: A novel hybrid cnn with a lstm-ae based framework. *Sensors*, 20(5), 1399.
- Kim, M., & Ramakrishna, R. (2005). New indices for cluster validity assessment. *Pattern Recognition Letters*, 26(15), 2353–2363.

- Kingma, D. P., & Welling, M. (2013, December). Auto-Encoding variational bayes.
- Ku, W., Storer, R. H., & Georgakis, C. (1995). Disturbance detection and isolation by dynamic principal component analysis. *Chemometrics and intelligent laboratory systems*, 30(1), 179–196.
- Kumar, A., & Mallipeddi, R. (2024). A deep clustering framework for load pattern segmentation. *Sustainable Energy, Grids and Networks*, 38, 101319.
- Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern recognition letters*, 42, 11–24.
- Lee, J., Kim, D., & Sim, S. (2024). Temporal multi-features representation learning-based clustering for time-series data. *IEEE Access*.
- Le Ray, G., & Pinson, P. (2019). Online adaptive clustering algorithm for load profiling. *Sustainable Energy, Grids and Networks*, 17, 100181.
- Liao, T. W. (2005). Clustering of time series data—a survey. *Pattern recognition*, 38(11), 1857–1874.
- Lin, J., Keogh, E., Lonardi, S., & Chiu, B. (2003). A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th acm sigmod workshop on research issues in data mining and knowledge discovery* (pp. 2–11).
- Ma, Q., Li, S., Zhuang, W., Wang, J., & Zeng, D. (2020). Self-supervised time series clustering with model-based dynamics. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9), 3942–3955.
- Ma, Q., Zheng, J., Li, S., & Cottrell, G. W. (2019). Learning representations for time series clustering. *Advances in neural information processing systems*, 32.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth berkeley symposium*

- sium on mathematical statistics and probability, volume 1: Statistics* (Vol. 5, pp. 281–298).
- Madiraju, N. S. (2018). *Deep temporal clustering: Fully unsupervised learning of time-domain features* (Unpublished master's thesis). Arizona State University.
- McLoughlin, F., Duffy, A., & Conlon, M. (2013). Evaluation of time series techniques to characterise domestic electricity demand. *Energy*, 50, 120–130.
- Mobtahej, P., Zhang, X., Hamidi, M., & Zhang, J. (2022, September). An LSTM-autoencoder architecture for anomaly detection applied on compressors audio data. *Computational and Mathematical Methods*, 2022, 1–22.
- Mrabah, N., Bouguessa, M., & Ksantini, R. (2020). Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift. *IEEE Transactions on Knowledge and Data Engineering*, 34(4), 1603–1617.
- Mrabah, N., Khan, N. M., Ksantini, R., & Lachiri, Z. (2019). Deep clustering with a dynamic autoencoder. *arXiv preprint arXiv:1901.07752*.
- Mrabah, N., Khan, N. M., Ksantini, R., & Lachiri, Z. (2020, October). Deep clustering with a dynamic autoencoder: From reconstruction towards centroids construction. *Neural Netw.*, 130, 206–228.
- Okereke, G. E., Bali, M. C., Okwueze, C. N., Ukekwe, E. C., Echezona, S. C., & Ugwu, C. I. (2023). K-means clustering of electricity consumers using time-domain features from smart meter data. *Journal of Electrical Systems and Information Technology*, 10(1), 2.
- Panapakidis, I. P., & Christoforidis, G. C. (2017). Implementation of modified versions of the k-means algorithm in power load curves profiling. *Sustainable Cities and Society*, 35, 83–93.
- Rajabi, A., Eskandari, M., Ghadi, M. J., Li, L., Zhang, J., & Siano, P. (2020).

- A comparative study of clustering techniques for electrical load pattern segmentation. *Renewable and Sustainable Energy Reviews*, 120, 109628.
- Rajabi, A., Li, L., Zhang, J., Zhu, J., Ghavidel, S., & Ghadi, M. J. (2017). A review on clustering of residential electricity customers and its applications. In *2017 20th international conference on electrical machines and systems (icems)* (pp. 1–6).
- Ren, Y., Wang, N., Li, M., & Xu, Z. (2020). Deep density-based image clustering. *Knowledge-Based Systems*, 197(1), 105841.
- Rezaei, M., Dorigatti, E., Ruegamer, D., & Bischl, B. (2021). Learning statistical representation with joint deep embedded clustering. *arXiv preprint arXiv:2109.05232*, 1.
- Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53–65.
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.
- Shi, X., et al. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems* (pp. 802–810).
- Sun, L., Zhou, K., & Yang, S. (2020). An ensemble clustering based framework for household load profiling and driven factors identification. *Sustainable Cities and Society*, 53, 101958.
- Tang, Y., Xie, Y., Yang, X., Niu, J., & Zhang, W. (2019). Tensor multi-elastic kernel self-paced learning for time series clustering. *IEEE Transactions on Knowledge and Data Engineering*, 33(3), 1223–1237.

- Tayeh, T., Aburakhia, S., & Myers, R. (2022). An attention-based ConvLSTM autoencoder with dynamic thresholding for unsupervised anomaly detection in multivariate time series. *arXiv preprint*. Retrieved from <https://arxiv.org/abs/2201.09172>
- Team, D. D. (2023). *Dask: Scalable analytics in python*. Retrieved from <https://www.dask.org/>
- Teeraratkul, T., O'Neill, D., & Lall, S. (2017). Shape-based approach to household electric load curve clustering and prediction. *IEEE Transactions on Smart Grid*, 9(5), 5196–5206.
- Tenenbaum, J. B., Silva, V. d., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500), 2319–2323.
- Trosten, D. J., Strauman, A. S., Kampffmeyer, M., & Jenssen, R. (2019). Recurrent deep divergence-based clustering for simultaneous feature learning and clustering of variable length time series. In *Icassp 2019-2019 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 3257–3261).
- UK Power Networks. (2014). *Smartmeter energy consumption data in london households, 23/11/2011-28/02/2014* (1st ed.). UK Power Networks. Retrieved May-2023, from <https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-l> ([dataset])
- Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., & Chervyakov, N. I. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177, 232–243.
- Vera-Vera, J. F., & Roldán-Nofuentes, J. A. (2024). A robust alternating least squares k-means clustering approach for times series using dy-

- namic time warping dissimilarities. *Mathematical Biosciences and Engineering*, 21(3), 3631–3651.
- Vercamer, D., Steurtewagen, B., Van den Poel, D., & Vermeulen, F. (2015). Predicting consumer load profiles using commercial and open data. *IEEE Transactions on Power Systems*, 31(5), 3693–3701.
- Verma, S., Singh, S., & Majumdar, A. (2021). Multi-label LSTM autoencoder for non-intrusive appliance load monitoring. *Electric Power Systems Research*, 199, 107414.
- Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on machine learning* (pp. 1096–1103).
- Wachowiak, M. P., Moggridge, J. J., & Wachowiak-Smolíková, R. (2019). Clustering continuous wavelet transform characteristics of heart rate variability through unsupervised learning. In *2019 41st annual international conference of the ieee engineering in medicine and biology society (embc)* (pp. 4584–4587).
- Wan, E. A., & Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the ieee 2000 adaptive systems for signal processing, communications, and control symposium (cat. no. 00ex373)* (pp. 153–158).
- Wang, C. (2017). A study of sufficient dimension reduction methods.
- Wang, X., Smith, K., & Hyndman, R. (2006). Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13, 335–364.
- Wang, Y., Chen, Q., Kang, C., Zhang, M., Wang, K., & Zhao, Y. (2015). Load profiling and its application to demand response: A review. *Tsinghua Science and Technology*, 20(2), 117–129.
- Wang, Y., Kong, Y., Tang, X., Chen, X., Xu, Y., Chen, J., et al. (2020).

- Short-term industrial load forecasting based on ensemble hidden Markov model. *IEEE Access*, 8, 160858–160870.
- Wang, Y., et al. (2015). Computational complexity of traditional clustering algorithms for smart meter data. *IEEE Transactions on Power Systems*, 30(4), 2345–2356.
- Wettschereck, D., & Dietterich, T. G. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19, 5–27.
- Xie, J., Girshick, R., & Farhadi, A. (2015, November). Unsupervised deep embedding for clustering analysis.
- Xie, J., Girshick, R., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International conference on machine learning* (pp. 478–487).
- Xu, C., Huang, H., & Yoo, S. (2021a). A deep neural network for multivariate time series clustering with result interpretation. In *2021 international joint conference on neural networks (ijcnn)* (pp. 1–8).
- Xu, C., Huang, H., & Yoo, S. (2021b, July). A deep neural network for multivariate time series clustering with result interpretation. In *2021 international joint conference on neural networks (IJCNN)*. IEEE.
- Yang, J., Parikh, D., & Batra, D. (2016). Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5147–5156).
- Yang, S.-L., Shen, C., & Zhaou, K.-L. (2013). A review of electric load classification in smart grid environment. *Renewable and Sustainable Energy Reviews*, 24, 103–110.
- Yang, W., Li, X., Chen, C., & Hong, J. (2022). Characterizing residential load patterns on multi-time scales utilizing lstm autoencoder and electricity consumption data. *Sustainable Cities and Society*, 84, 104007.

- Yilmaz, S., Chambers, J., & Patel, M. K. (2019). Comparison of clustering approaches for domestic electricity load profile characterisation-implications for demand side management. *Energy*, 180, 665–677.
- Yin, J., & Yang, Q. (2005). Integrating hidden markov models and spectral analysis for sensory time series clustering. In *Fifth ieee international conference on data mining (icdm'05)* (pp. 8–pp).
- Zakaria, J., Mueen, A., & Keogh, E. (2012). Clustering time series using unsupervised-shapelets. In *2012 ieee 12th international conference on data mining* (pp. 785–794).
- Zarabie, A. K., Lashkarbolooki, S., Das, S., Jhala, K., & Pahwa, A. (2019). Load profile based electricity consumer clustering using affinity propagation. In *2019 ieee international conference on electro information technology (eit)* (pp. 474–478).
- Zhang, Y., Huang, T., & Bompard, E. F. (2018). Big data analytics in smart grids: a review. *Energy informatics*, 1(1), 1–24.
- Zhong, Y., Huang, D., & Wang, C.-D. (2023). Deep temporal contrastive clustering. *Neural Processing Letters*, 55(6), 7869–7885.
- Zhou, W., & Zhou, Q. (2019). Deep embedded clustering with adversarial distribution adaptation. *IEEE Access*, 7, 113801–113809.
- Zou, J., Li, Q., Yang, S., Bai, H., & Zheng, J. (2017). A prediction strategy based on center points and knee points for evolutionary dynamic multi-objective optimization. *Applied Soft Computing*, 61, 806–818.

ملخص الدراسة

يُمثل النمو السريع لبيانات السلاسل الزمنية في مختلف المجالات كمجال الطاقة، التمويل، الرعاية الصحية والنقل تحديات تحليلية كبيرة نظراً لارتفاع أبعادها، وتبعياتها الزمنية، ومحدودية البيانات المُصنّفة. غالباً ما تُواجه أساليب التجميع التقليدية صعوبات في التعامل مع هذه التعقيدات، مما يؤدي إلى تحديد غير فعال للتجمعات الأساسية. ولمعالجة هذه المشكلات، أصبحت أساليب التجميع العميق، وخاصةً تلك التي تستخدم المُرمّزات التلقائية، شائعة الاستخدام، فهي تُقلّل الأبعاد بشكل فعال وتستخرج السمات عن طريق ضغط البيانات عالية الأبعاد في تمثيلات كامنة مضغوطة، مع التقاط الأنماط الأساسية وتقليل الضوضاء. وينتج عن ذلك تجميع أكثر متانة وقابلية للتطوير لمجموعات بيانات السلاسل الزمنية المعقدة. ومع ذلك، تواجه أساليب التجميع العميق تحديات تتعلق بعشوائية السمات (FR) وانحراف السمات (FD)، والتي قد تُزعزع استقرار السمات المكتسبة وتُغيّر التمثيلات أثناء التدريب، مما يؤثر سلباً على جودة التجميع. وعلى الرغم من أن النماذج الحديثة حاولت الموازنة بين عشوائية السمات (FR) وانحراف السمات (FD)، إلا أنها لم تُعالج بشكل كافٍ الخصائص المحددة لمجموعات بيانات السلاسل الزمنية. لتحقيق التوازن الفعال بين FR وFD، تقترح هذه الأطروحة نموذجين مبتكرين للتجميع المضمن العميق والمتسلسل لتجميع السلاسل الزمنية: المشفر التلقائي الديناميكي للذاكرة طويلة المدى قصيرة المدى (LSTM-DynAE) والمشفر التلقائي الديناميكي التلافيفي للذاكرة طويلة المدى قصيرة المدى (ConvLSTM-DynAE). يدمج نموذج LSTM-DynAE مشفرات LSTM التلقائية مع دالة خسارة ديناميكية تُحوّل تركيز التدريب تدريجياً من إعادة البناء إلى التجميع، مع الحفاظ على تمثيلات مستقرة وتمييزية. يوسّع نموذج ConvLSTM-DynAE هذا الإطار من خلال دمج مشفرات LSTM التلقائية التلافيفية (ConvLSTM)، مما يُتيح التعلم المتزامن للخصائص المكانية والزمانية، وهو أمر مفيد بشكل خاص لتحديد أنماط الاحتمال في الشبكات الذكية. تم التحقق من صحة النماذج المقترحة بدقة على كل من مجموعات البيانات المرجعية المُصنّفة ومجموعات بيانات العدادات الذكية الواقعية. حقق LSTM-DynAE دقةً مذهلةً (Acc) بلغت 0.97، ومعلوماتٍ طبيعية متبادلةً (NMI) بلغت 0.95 في مجموعة بيانات Heartbeat، مُظهرًا أداءً قوياً في التجميع. كما تفوق ConvLSTM-DynAE في تحديد أنماط الاحتمال، محققاً درجات (Silhouette (Sil بلغت 0.980 في مجموعة بيانات لندن و0.996 في مجموعة بيانات أيرلندا، مُميزاً بذلك أنماط استهلاك الطاقة لدى المستهلكين بفعالية. تُؤكد هذه النتائج فعالية تقليل الأبعاد العميق القائم على المُرمّز التلقائي واستخراج الميزات في تمكين تجميع سلاسل زمنية دقيقة وقوي. ومن خلال مُعالجة تحديات FR وFD بنجاح من خلال دوال الخسارة الديناميكية، يُطور البحث أحدث التقنيات في التجميع الزمني، ويُوفر أدواتٍ عمليةً لتحليلات الشبكات الذكية وإدارة الطاقة.

جامعة البحرين

كلية تقنية المعلومات



أساليب جديدة لخفض الأبعاد تعتمد على التشفير التلقائي للتجميع الفعّال

أطروحة مقدمة كجزء من متطلبات الحصول على
درجة الدكتوراة في علوم الحوسبة والمعلومات

إعداد

حسناء جعفر آل سعيد

٢٠٠١١٩٣٥

إشراف

أ.د.رياض قسطنطيني

(أستاذ مشارك)

أ.د.فيصل الخطيب

(أستاذ)

جامعة البحرين

مملكة البحرين

يونيو ٢٠٢٥