Privacy-preserving and Learning-efficient Energy Data Analysis for Advanced Metering Infrastructure

YU HE

B.E., M.E. in Electrical Engineering

Supervisor: Dr Fengji Luo
Associate Supervisor: Prof. Gianluca Ranzi

A thesis submitted in fulfilment of the requirements for
the degree of Doctor of Philosophy

School of Civil Engineering
Faculty of Engineering
The University of Sydney
Australia

2025

Abstract

The deployment of Advanced Metering Infrastructure has transformed modern power systems by facilitating the collection of detailed, real-time energy consumption data. This granular data can support a variety of upper-level demand-side management applications, such as building/home energy management, electricity retail pricing, and microgrid operational planning. Effective implementation of these applications requires precise knowledge of both current and future load patterns. Consequently, two analytical techniques have emerged: short-term load forecasting and load super-resolution. The former predicts future energy consumption patterns in smart grid operations, and the latter reconstructs high temporal-resolution load profiles from low-resolution or incomplete data. Currently, the deep learning approach provides state-of-the-art results for both techniques due to its powerful ability to capture complex nonlinear relationships among the energy load profiles, enabling significant enhancements in prediction accuracy and reconstruction fidelity.

Despite significant advancements in deep learning methods for both short-term load forecasting and load super-resolution, their translation into practical deployment remains constrained by several limitations. Individual households or small-scale facilities typically lack sufficient training data to support effective model training. While sharing and aggregating raw data across users can lead to privacy concerns for individuals. Deep learning methods often generalize poorly. For example, load super-resolution methods rely on high-resolution data for training, and one model can only support a single scaling factor, i.e., a fixed ratio between low- and high-resolution data.

Based on the above challenges, the thesis is dedicated to mitigating the limitations in deploying deep learning techniques in energy systems. To this end, my investigation centers on two foundational tasks within advanced metering infrastructure data analytics: short-term load forecasting and energy load super-resolution. The first research of this thesis proposes a privacy-preserving short-term forecasting framework

specifically designed for residential energy users. This framework allows households to collaboratively train load forecasting models without sharing their consumption data. The framework uses a decentralized clustering method to group users by energy consumption patterns, followed by a decentralized federated learning to train models. An asynchronous communication protocol ensures resilience against real-world communication delays and interruptions.

The second research proposes an unsupervised load super-resolution system that eliminates the reliance on high-resolution training data. Instead of requiring paired low- and high-resolution datasets for training the model, the proposed system is trained solely in low-resolution load data. In addition, the system incorporates a noise identification module that automatically detects and adapts to different types of noise in the load signals, improving reconstruction accuracy.

The third research addresses the poor generalization of deep learning methods through two separate works. The first introduces a meta learning-based short-term load forecasting framework that trains a generic model that quickly adapts to individual users who only have a limited volume of training data. The second work proposes a load super-resolution approach capable of reconstructing load profiles at arbitrary temporal resolutions from a single training instance. This enables efficient deployment of resource-constrained devices and greatly enhances model flexibility.

Comprehensive numerical experiments, case studies, and ablation tests based on several real-world datasets are conducted to validate the effectiveness of the proposed methods.

Acknowledgements

I would like to express my deepest gratitude to my principal supervisor, Dr. Fengji Luo, and my associate supervisor, Professor Gianluca Ranzi, for their invaluable guidance and consistent support throughout my PhD journey. I especially thank Dr. Fengji Luo, whose rigorous academic standards and personal dedication have left a profound impact on my development. I will never forget the night before my journal paper submission deadline, when he patiently helped me revise the manuscript until nearly midnight despite having a high fever of 39°C. His commitment and attention to detail shaped the quality of my research and instilled in me the mindset of a true scholar.

I am also sincerely grateful to my girlfriend, Yuanyuan Liu, whose unwavering support and companionship carried me through this long journey. Over the past four years, she has endured the challenges of a long-distance relationship between Sydney and Brisbane with strength and patience, always finding time to visit and be by my side whenever possible. Her presence and encouragement during difficult moments gave me emotional resilience to keep moving forward.

I am also deeply thankful to my colleagues and the staff at the University of Sydney for their friendship and ongoing support. Their kindness and collaboration made my PhD journey productive and genuinely enjoyable.

I would also like to thank the University of Sydney for providing access to the research facilities, computational infrastructure, and academic environment that made this work possible. I am particularly grateful for the full scholarship I received, which allowed me to pursue this research without financial burden and allowed me to dedicate myself fully to my academic goals.

List of Publications

Peer-referred journal articles:

(1) **Y. He**, F. Luo, and G. Ranzi, "Privacy-preserving and hierarchically federated framework for short-term residential load forecasting", *IEEE Transactions on Smart Grid*. pp. 1-13, 2023.

(2) **Y. He**, F. Luo, S. Kanhere and G. Ranzi, "Unsupervised load super-resolution based on contrastive learning", *IEEE Transactions on Smart Grid*, pp. 1-12, 2025.

(3) **Y. He**, F. Luo, and G. Ranzi, "Load reconstruction with arbitrary super resolutions", *IEEE Transactions on Power Systems*. pp. 1-3, 2023.

(4) **Y. He**, F. Luo, and G. Ranzi, "Transferrable model-agnostic meta-learning for Short-term household load forecasting with limited training data," *IEEE Transactions on Power System.* vol. 37, pp. 3177-3180, 2022.

(5) Z. Zhao, F. Luo, **Y. He**, and Gianluca Ranzi, "Personalized P2P energy trading system based on socio-demographic characteristic inference and AC network constraints," *Applied Energy*. vol. 368, pp. 123333.

(6) B. Zhang, F. Luo, **Y. He**, and G. Ranzi, "GT-NILM: a generative, transferable non-intrusive load monitoring framework based on conditional diffusion model and domain adaptation," *IEEE Transactions on Consumer Electronics*.

Full papers in conferences proceedings:

(7) Z. Ye, **Y. He**, and F. Luo, "Evaluation study of deep learning-based models on probabilistic power load forecasting," *in Proc. 2023 IEEE Conference on Energy Internet and Energy System Integration*, 2023, Hangzhou, China. **(Best Paper Award)**

(8) B. Zhang, F. Luo, and **Y. He**, "Evaluation of diffusion models on non-intrusive load monitoring," *in Proc. 2023 IEEE Conference on Energy Internet and Energy System Integration.* 2023, Hangzhou, China, 2023.

(9) **Y. He**, F. Luo, and G. Ranzi, "Comparison study of collaborative learning techniques on residential short-term load forecasting", *in Proc. 2022 IEEE Sustainable Power and Energy Conference*, 2023, Perth, Australia, 2022.

(10) **Y. He**, F Luo, G. Ranzi, and W. Kong, "Short-term residential load forecasting based on federated learning and load clustering," *in Proc. 2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, Aachen, Germany, 2021.

(11) **Y. He**, F. Luo, L. Yip, and G. Ranzi, "Short-term power load forecasting based on distilled temporal convolutional networks," in *Proc. 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events*, Biarritz, France, 2024.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Yu He (Signature)

August 2025

Authorship Attribution Statement

This thesis contains the following materials:

(1) **Y. He**, F. Luo, and G. Ranzi, "Privacy-preserving and hierarchically federated framework for short-term residential load forecasting", *IEEE Transactions on Smart Grid*. pp. 1-13, 2023. This publication is Chapter 3 of this thesis.

(2) **Y. He**, F. Luo, S. Kanhere and G. Ranzi, "Unsupervised load super-resolution based on contrastive learning", *IEEE Transactions on Smart Grid*, pp. 1-12, 2025. This publication is Chapter 4 of this thesis.

(3) **Y. He**, F. Luo, and G. Ranzi, "Load reconstruction with arbitrary super resolutions", *IEEE Transactions on Power Systems*. pp. 1-3, 2023. This publication is Chapter 5 of this thesis.

(4) **Y. He**, F. Luo, and G. Ranzi, "Transferrable model-agnostic meta-learning for Short-term household load forecasting with limited training data," *IEEE Transactions on Power System.* vol. 37, pp. 3177-3180, 2022. This publication is Chapter 5 of this thesis.

(5) **Y. He**, F. Luo, and G. Ranzi, "Comparison study of collaborative learning techniques on residential short-term load forecasting", *in Proc. 2022 IEEE Sustainable Power and Energy Conference*, 2023, Perth, Australia, 2022. Part of this publication is in Chapter 3 of this thesis.

(6) **Y. He**, F. Luo, L. Yip, and G. Ranzi, "Short-term power load forecasting based on distilled temporal convolutional networks," in *Proc. 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events*, Biarritz, France, 2024. Part of this publication is in Chapter 3 of this thesis.

In addition to the statements above, in cases where I am not the corresponding author of the published item, permission to include the published material has been granted by the corresponding author.

Yu He (Signature)

August 2025

As a supervisor for the candidate upon whom this thesis is based, I can confirm that the authorship attribution statement above is correct.

Fengji Luo, (Signature)

August 2025

List of Abbreviations

AFL	Asynchronous federated learning
AMI	Advanced metering infrastructure
ANN	Artificial neural network
ARIMA	Autoregressive integrated moving average
AL	Aggregated learning
CH	Cluster head
CNN	Convolutional neural network
EDSR	Enhanced deep super-resolution
EFB	Embedding fusion block
EFG	Embedding fusion group
EFM	Embedding fusion module
ES	Exponential smoothing
FCN	Fully connected network
FCE	Frequency component error
FL	Federated learning
GAN	Generative adversarial network
GN	Global node
GRU	Gated recurrent unit
GS	Gradient sharing
IL	Individual learning
KNN	*k*-nearest neighbor
LR	Linear regression
LRM	Local regression mechanism
LSR	Load super-resolution
LSTM	Long short-term memory
MAC	Multiply-accumulate operation

MAE	Mean absolute error
MAML	Model-agnostic meta-learning
MAPE	Mean absolute percentage error
MGM	Mean gradient magnitude
MSE	Mean squared error
InfoNCE	Information noise contrastive estimation
PCA	Principal component analysis
RF	Random forest
RNN	Recurrent neural networks
RMSE	Root mean square error
RMSProp	Root mean square propagation
STLF	Short-term load forecasting
SVM	Support vector machine
TM	Tree-based method
WCSS	Within cluster sum of squares

Artificial Intelligence

During the preparation of the thesis the author used ChatGPT (see webpage at https://chatgpt.com) for the purposes of text enhancement. This includes paraphrasing, sentence structure, spelling. The author confirms that where text was modified by generative AI, the content was reviewed for possible errors, inaccuracies, and bias. The author takes full responsibility for the submitted thesis and ensures the work is their own and has used generative AI within the parameters of use (refer to the University of Sydney generative AI guide for researchers).

Yu He (Signature)

August 2025

Australian Government Support

I acknowledge the financial support received through the University of Sydney International Scholarship (SC3757), Postgraduate Research Support Scheme 2025, and the Postgraduate Research Scholarship in Machine Learning for Prefabricated Composite Building Components and Connections (SC4235).

List of Figures

List of Tables

Contents

CHAPTER 1
Introduction

In this thesis, two fundamental tasks in Advanced Metering Infrastructure (AMI) data analytics: short-term load forecasting and energy load super-resolution. Both represent essential components for enabling accurate, fine-grained, and intelligent energy management in modern power systems. This chapter presents the background, motivations, and the proposed approach that builds upon these two AMI data analyses.

1.1 Motivation and Background

The deployment of AMI has fundamentally transformed modern power systems from traditional one-way energy delivery networks into intelligent, data-driven ecosystems [1]. This evolution has enabled the collection of detailed, real-time energy consumption data, creating a large volume of useful information that serves as digital assets for energy systems. For example, these energy consumption data can support various demand-side management applications, including facilitating home energy management systems [2], enabling dynamic electricity retail pricing mechanisms [3], and helping with microgrid operational planning strategies [4].

However, the effective implementation of these advanced demand-side applications requires precise users' knowledge of both current consumption patterns and accurate predictions of future load behaviors. To meet these needs, Short-Term Load Forecasting (STLF) techniques are used to predict users' energy consumption over short horizons (i.e., from a few minutes to days), which helps the energy system to facilitate real-time market operations [5] or support demand response activation [6]. While Load Super Resolution (LSR) techniques are used to generate high time resolution of energy load data when only low-time resolution data are available, which enhances the power grid observability without requiring costly metering infrastructure upgrades [7].

In recent years, deep learning methods have achieved remarkable success in both STLF and LSR techniques [8], consistently outperforming conventional methods. Focusing first on STLF, deep learning-based methods such as recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) models [9], have been widely adopted in the research community. Compared with conventional methods, those RNN models have better capability to capture temporal patterns and long-term trends existing in energy load profiles [10]. In terms of LSR techniques, models such as Convolutional Neural Networks (CNNs) [11] and Generative Adversarial Networks (GANs) [12], have been employed to reconstruct high-resolution load profiles. Because these models are trained on large datasets, they generally achieve better reconstruction of energy load profiles compared to non-deep learning-based methods. Deep learning–based methods generally provide more accurate reconstructions of energy load profiles, particularly in regions with abrupt transitions or peak values.

1.2 Problem Statement and Objective

Despite the demonstrated effectiveness of deep learning-based STLF and LSR techniques, their practical deployment in smart grid environments faces significant methodological and operational challenges as follows.

- **A deep learning model's performance largely depends on the quality of the training data. Yet in practice, the amount of data is often limited due to various reasons, and privacy regulations restrict data sharing with other utilities.** Specifically, when single residential users apply their own load data to train deep learning models and there is no sufficient historical load data, the overfitting issue [13] might occur, which degrades the performance of learning models. One possible solution is to perform model training process by aggregately the load data of multiple users that would usually require the collection of the multi-user load data to a central server with the associated privacy concerns for the users. An attacker could easily infer household

occupancies by eavesdropping on the users' load data in the data transmission process.

- **Most existing supervised learning for LSR requires paired data, where low-resolution load data serve as inputs and their corresponding high-resolution load data serve as outputs. While constructing such pairs, a large volume of high-resolution load data must first be collected.** In real-world settings, however, only coarse-grained measurements of load data are typically accessible, which makes supervised learning of LSR impractical. Obtaining such high-resolution measurements often require costly metering infrastructure upgrades, which defeats the primary purpose of LSR technique that aims to enhance the time resolution of energy data without costly hardware upgrades.
- **In industrial applications, STLF and LSR techniques often exhibit poor generalization, as models trained under specific conditions may fail to adapt to other users, region, or operational environments.** Consider the scenario of deploying an STLF model in a newly occupied household or a short-term rental with limited historical usage records. In such cases, the model training could be performed by using data from other sources. However, the energy consumption patterns of the target user can differ markedly from those represented in the training data. In addition, most existing deep learning-based LSR models are limited to upscaling energy load profiles at a single, fixed time resolution. For example, an LSR model trained to upscale from 5-minute to 1-minute intervals cannot generalize 15-minute to 5-minute or 30-minute to 10-minute, while retraining separate models for every resolution is computationally inefficient and infeasible.

1.3 Summary of Contributions

In response to the aforementioned challenges in LSTM and LSR tasks, the key contributions of this thesis are delineated as follows.

1.3.1 Privacy-preserving Load Forecasting for Residential Users

To address training data shortage and user privacy challenges in existing STLF methods, this research proposes a novel federated learning framework that enables accurate and privacy-preserving short-term load forecasting. This framework performs the privacy-preserving through two stages. On the first stage, the systems perform federated user clustering techniques to group households of similar energy load patterns into different clusters. The federated clustering technique ensures the system does not share energy load data. On the second stage, the systems apply a hierarchically federated learning strategy that performs STLF model training both within and across clusters. In this stage, the actual load data are not communicated, but only the model parameters are transferred, allowing localized collaboration while leveraging global information. To ensure robustness in residential settings, an asynchronous communication protocol is integrated. This protocol enables federated model training to proceed without requiring strict synchronization, effectively handling user disconnections and high-latency communication. This can improve the fault tolerance of model transmission and adaptability under unreliable network conditions. Together, these innovations enable a secure, efficient, and high-performance STLF framework across large-scale user populations.

1.3.2 Mitigating the Unavailability of High-resolution Load Data

This research proposes a novel LSR system that enables LSR model training using only low-resolution measurements from smart meters, without requiring any high-resolution data. The system applies self-supervised learning to extract meaningful representations from low-resolution data. This is done by down-scale the low-resolution data into lower-resolution data to form pairs for LSR model training. A contrastive learning mechanism is proposed to identify structurally similar load instances, enabling the model to produce high-quality reconstructions even under substantial noise interference in the model inputs. This is the first study to demonstrate

that unsupervised LSR can achieve performance comparable to or exceeding supervised methods, in the absence of high-resolution ground truth.

1.3.3 Enhance the Adaptability of Learning Models

To improve the learning model's adaptability to varying practical contexts, the research is organized into two individual studies. The first study introduces an LSR technique that, for the first time, enables accurate reconstruction of load data at arbitrary temporal resolutions using a single trained model. Unlike existing LSR methods that are limited to fixed up-sampling ratios. During model training, both low- and high-resolution load data are mapped onto a shared one-dimensional coordinate system, with each data point assigned a unique temporal coordinate. The model is then trained on load values and their corresponding coordinates, allowing it to learn a continuous mapping over time. As a result, the trained model can generate load estimates at any desired resolution by querying points in the coordinate space, achieving high reconstruction accuracy across varying granularities without retraining.

The second study addresses limited training data in individual residential settings; this study proposes a Transferable Model-Agnostic Meta-Learning (T-MAML) framework for STLF tailored to single households. The system first conducts a meta-training, where data from multiple households are used to train a generalized model with transferable model parameters. This is followed by a meta-transfer stage, where the generic model is efficiently fine-tuned using a small amount of target household data, enabling personalized STLF without extensive computation or communication overhead at the user-side. Unlike federated learning approaches, the proposed method eliminates the need for iterative communication rounds and is computationally lightweight on the household side. Moreover, the T-MAML framework incorporates both meta-training and meta-transfer mechanisms, allowing it to capture both global load patterns across households and household-specific consumption characteristics, thereby offering a more practical and adaptive solution for real-world residential

forecasting applications.

1.4 Thesis Outline

This thesis is structured into seven chapters and Chapter 1 provides the background and motivation for this thesis and outlines the primary contributions. Chapter 2 presents a comprehensive review of related work, highlighting the current state of the art.

Chapter 3 investigates the problem of STLF at the residential user level. Households' energy load data cannot be shared due to privacy concerns, and communication between devices may suffer from delays, message loss, or user dropout typical in residential networks. Unlike conventional forecasting setups that assume centralized data access and stable infrastructure, this work considers a decentralized environment where individual users possess limited historical load data, are unwilling to share raw data due to privacy concerns, and operate over unreliable residential communication networks. The study explores how collaborative forecasting can still be achieved under these real-world limitations.

Chapter 4 investigates the task of unsupervised LSR in residential energy systems, where high-resolution data are unavailable, and the collected low-resolution measurements may be noisy, irregular, or incomplete. The scenario considered reflects practical constraints in real-world deployments: smart meters may report data at coarse intervals due to bandwidth or storage limitations, historical fine-grained data may not exist for new users or legacy systems, and the collected data often exhibits diverse noise patterns caused by sensing errors or environmental disturbances. The work explores how to reconstruct accurate high-resolution load profiles from such limited and noisy observations without relying on supervised learning or clean high-resolution ground truth.

The last research work addresses the adaptability of the deep learning model through two different studies. The first one in Chapter 5 is the LSR in residential smart meter settings, for which existing methods require retraining the LSR model for each

desired resolution, making them impractical for multi-resolution applications. This work introduces an LSR technique capable of reconstructing load profiles with arbitrary target resolutions using a single trained model. The system is particularly suited to edge devices and adaptive energy management tasks in smart grid environments. The second research work in Chapter 6 is to address the challenge of STLF at the single-household level, where only a limited amount of historical load data is available and the computational and communication resources are constrained. The study focuses on a typical residential scenario involving newly connected or transient users, such as those in short-term accommodations or recently relocated households, where conventional and federated learning methods often fail due to data sparsity and high infrastructure demands.

Chapter 7 presents the concluding remarks of this thesis and outlines potential directions for future research.

CHAPTER 2

Literature Review

This chapter presents a comprehensive review of STLF, covering its principles, representative methods, and recent advancements. It then provides an in-depth discussion of LSR, including its technical challenges and state-of-the-art solutions.

2.1 Review for Short-term Load Forecasting (STLF)

STLF refers to the techniques that predict the electricity demand over a few minutes up to a few hours. Because modern power systems lack large-scale energy storage, generation must be continuously balanced with forecasted demand in real-time. A well-designed STLF framework, when effectively integrated into modern power system operations, can deliver substantial economic, environmental, and reliability benefits to both the electricity sector and the community. Even marginal enhancements in STLF accuracy can translate into substantial economic gains. For example, a minor 1% reduction in forecasting error has been estimated to yield annual savings of approximately £10 million for the United Kingdom's electricity grid [14]. To the best of the author's knowledge, the study of STLF can be traced back to the era of the 1960s. A key challenge in this field is that short-term energy load exhibit high volatility and strong dependency on human behavioural patterns [15]. When early approaches primarily rely on linear regression and some basic time series models [16, 17]. With the emergence of machine learning techniques, STLF has experienced a significant leap in terms of forecasting accuracy. Motivated by this, the thesis categorizes existing STLF techniques and approaches into two major classes: statistical-based and machine learning-based STLF.

2.1.1 Statistical-based STLF

Statistical-based STLF methods can be further categorized into two groups: time series models and state space models. Although hybrid approaches combining different model types have been widely developed, the definition boundaries between the two categories remain clear. Time series models primarily rely on historical load data to predict future values. While state-space models employ recursive estimation techniques to iteratively refine forecasts as new observations become available.

2.1.1.1 Time Series-based models for STLF

The time series-based models generally assume that load values are statistically influenced by preceding observations and that the power consumption data show periodic patterns, ranging from intra-day to seasonal cycles. Those models broadly include linear regression, Autoregressive Integrated Moving Average (ARIMA), and Exponential Smoothing (ES).

Linear regression methods [18-20] represent one of the earliest and most widely adopted approaches in STLF, aiming to capture the relationship between energy load values and some other external variables such as temperature, time of day, and calendar effect. These methods assume that there exists a linear correlation between the load values and external variables. However, this assumption often fails to accurately reflect actual load behaviors. Hence, Generalized Linear Models (GLMs) [21-23] have been introduced. Unlike linear regression methods, GLMs are designed to model complex load patterns and can effectively handle load data exhibiting skewness, heavy tails, or heteroscedasticity. GLMs specify appropriate link functions, which define the relationship between the linear predictor and past historical load data, aiming the model to accommodate diverse distributional characteristics of load data.

ARIMA consists of multiple components, each targeting a specific characteristic of energy load data. An autoregressive component for predicting the future energy load using past observations. It uses an integration component to remove trends from energy

load data, thereby making the time series more stationary, and a moving average component to adjust forecast errors. Foundational work such as [24] aims to design a more robust autoregressive component to capture regular short-term energy load patterns, such as daily and weekly cycles. [25] introduces external information into STLF, integrating temperature data as an exogenous variable to allow the model to consider energy load data as weather-driven load data. Due to the model's structural simplicity and ease of implementation, ARIMA-based STLF methods are often incorporated with other techniques to form hybrid forecasting frameworks. For example, [26, 27] combines ARIMA with wavelet decomposition and fuzzy inference systems. The methods decompose energy data with different time frequency bands and use ARIMA to forecast each frequency band individually. [28] designs an online information network that enables the model to adjust forecasts in response to incoming energy load data, thereby allowing ARIMA to perform real-time STLF. Some machine learning techniques have also been integrated with ARIMA within hybrid forecasting frameworks. For example, ARIMA combined with support vector machines [29-31], LSTM [32-34], and feedforward neural networks [35-37]. Experimental results demonstrate that such hybrid frameworks achieve significantly higher forecasting accuracy compared with using ARIMA alone for STLF.

ES methods are also highly simple and computationally efficient, making them particularly suitable for real-time and resource-constrained STLF applications. The core idea of ES is to assign exponential weights to historical load values, giving more importance to recent observations while still accounting for the influence of past data. Existing research on ES-based STLF can be broadly categorized into several directions. One line of research focuses on modifying algorithmic rules to assign more selective weights to particularly important load values, enhancing sensitivity to relevant load patterns while down-weighting noise or outliers [38-41]. Another set of studies incorporates external inputs, such as binary indicators for holidays, as exogenous variables. The ES-based STLF aims to capture correlations between electricity demand and those external conditions [42]. A third branch of research aims to decompose the

energy load data into different sets and process them individually with ES methods. For example, [43] decomposes energy load data into level, trend, and seasonality. Similar to ARIMA-based STLF methods, due to the ES method's low complexity, it has been integrated into hybrid forecasting frameworks, for example, including combinations with moving averages [44], ANNs [45, 46], and RNNs [47-49].

2.1.1.2 State Space Models for STLF

State space models in STLF forecast future energy load in a recursive manner, where the prediction at time is generated from past observations and previous forecasts. In contrast, other statistical-based STLF approaches compute forecasts for any future time directly from historically observed variables without relying on prior predicted values. The State-space methods use two major equations. The state equation computes and maintains a vector of latent variables called state. It contains information such as long-term trends and seasonal effects on historical load values from early time steps. An observation equation links such a state to actual load data at any given time for forecasting. This framework allows us to describe STLF in a Markovian fashion: the current state of the model is only related to the prior state and the latest observation, eliminating the need to reprocess the full historical data. Typical examples of state-space models in STLF include Kalman Filter-based STLF [50-52], Extended Kalman Filter-based STLF [53-55], the Unscented Kalman Filter-based STLF [56, 57], Particle Filter-based STLF [58], and Bayesian dynamic models for STLF [59-61].

2.1.2 Machine Learning-based STLF

Machine learning-based methods for STLF can broadly be categorized into classical machine learning techniques and deep learning methods. The primary difference lies in model scale, where classical techniques generally involve far fewer trainable model parameters compared to deep learning techniques. While their

operational principle is similar: learning a mapping from historical load data to future load values.

2.1.2.1 Classical machine learning models for STLF

Owing to classical machine learning methods' limited model capacity, they often rely on large feature engineering (e.g., selecting proper weather inputs, calendar features, etc.). Classical machine learning methods in STLF can be broadly classified into many categories, where two major ones are Tree-based Methods (TMs) and Support Vector Machines (SVMs) methods.

In the early 2010s, TMs emerge as a popular choice for STLF due to their robustness and ability to model complex nonlinear relationships. These methods can naturally handle multiple types of inputs, such as numerical load values, binary holiday flags, and categorical calendar indicators, without the need for extensive preprocessing and normalization. Random Forest (RF) methods are one type of TMs [62], which builds a set of decision trees [63]. Each tree is trained on a portion of the training data, and the final forecast is obtained by averaging the predictions of all trees. Representative RF-based STLF methods can be classified into certain research directions: (1) The studies that focus on feature engineering, which involves identifying the most relevant input variables and transforming them into meaningful embeddings to enhance the performance [64-66]. (2) The studies that work on improving the hyperparameter optimization, which employs techniques such as grid search and Bayesian optimization to fine-tune model performance [67]. (3) And the studies construct hybrid modeling, which integrates tree-based algorithms with other techniques [68, 69].

Another type of classical machine learning method in STLF that is worth mentioning is SVMs. Its research trajectory is like that of RF-based STLF methods, initially focusing on improvements to its core algorithm design. For example, some early studies aim to design a more effective kernel function to improve the ability of

SVMs to capture nonlinear relationships in load patterns. A kernel function is an essential component in SVMs that maps input data into higher-dimensional embeddings [70]. The study [71] proposes a Gaussian wavelet kernel that decomposes energy load data into multiple time-frequency components and applies SVMs individually to each component. Studies in [72, 73] employs a multiple-kernel SVMs for STLF, which utilizes three different kernel functions to process the energy load data and apply the most suitable one for forecasting each specific energy load profile. Along with the development of kernel functions in SVMs, some studies focus on designing a hybrid architecture that can further boost the performance of STLF, such as combining SVMs with ARIMA [74], with particle swarm optimization [75], with wavelet decomposition [76], or with feature selection schemes [77].

2.1.2.2 Deep Learning models for STLF

Deep learning methods use a multi-layer neural architecture, where each layer consists of many interconnected computational units called neurons. Each neuron contains basic trainable weights, often referred to as the model's parameters, which are updated and trained via the backpropagation method to minimize a pre-defined loss function. Deep learning is termed "deep" because it typically employs a large number of stacked layers, resulting in highly complex model architectures. Such depth enables to include a vast number of neurons. Those neurons collectively can represent highly nonlinear and abstract features. Early studies of deep learning-based STLF aim to determine whether deep learning methods could achieve higher forecasting accuracy than conventional ones. Representative studies include [78] and [79], which compare deep neural networks against shallow neural networks and other conventional machine learning models for STLF. Researchers also investigate the application of CNNs in STLF. CNNs are originally developed for image processing and can only accept the two-dimensional data format as inputs. Some studies transform time series data into a two-dimensional representation and directly input them into the CNNs for model

training [80, 81]. Others design one-dimensional convolutional layers, which are then stacked to construct one-dimensional CNN architectures capable of directly modelling from raw sequential load data [82-84]. These investigations provide empirical evidence that deeper architectures can offer performance gain in STLF.

Later studies of deep learning-based STLF methods shift from comparing alternative deep learning methods to investigating the modelling of long- and short-term temporal dependencies in energy load profiles. In energy load data, long-term dependencies refer to patterns such as seasonal or weekly consumption trends, whereas short-term dependencies are the rapid fluctuations driven by recent load behaviours. RNNs have been widely employed for STLF [85, 86], along with their variants such as LSTM [87-91] and Gated Recurrent Unit (GRU) networks [92, 93] in STLF. These architectures incorporate gating mechanisms and memory cells in their design, enabling them to selectively retrain, update, and forget information over time. Hence, this design allows RNN-based models with a superior capability to capture long-term temporal dependencies in energy load data. Subsequently, research on deep learning-based STLF increasingly focuses on hybrid architectures that combine a few techniques together. The study [94] combines an LSTM and GRU network, where LSTMs are used to capture long-term dependencies of energy load data, while GRUs are employed to model short-term variations with reduced computational overhead. Or the work [95] integrates LSTM and CNN frameworks to leverage convolutional layers for local feature extraction prior to LSTM processing. Other studies apply conventional machine learning methods, such as principal component analysis (PCA) [96] or RF-based feature selection [97], as a preprocessing step before feeding the data into LSTMs. These hybrid approaches aim to explore the performance limits of model integration.

2.2 Load Super-resolution (LSR)

Energy load profiles are typically obtained from smart meters, which can record energy consumption at intervals between a few seconds and up to 60 minutes. High time

resolution load data refers to time-stamped energy measurements acquired sufficiently short, typically sub-minute to a few minutes. Such fine-grained data can offer significant benefits for smart grid applications; however, its large volume of data could burden storage systems, cloud infrastructure, and processing pipelines. This can lead to higher operational costs for utilizers and building owners. According to [98, 99], one million smart meters recording data at half-hour intervals can generate approximately 1,460 terabytes of data annually. In comparison, 135 million smart meters in the United States can produce up to 54 petabytes of data per year [100]. Hence, LSR has emerged as a potential solution to mitigate such storage burden by reconstructing high temporal resolution load profiles from measurements that are low-resolution and coarse ones. To facilitate a structured understanding of LSR methods, this thesis classifies them into two main categories: non-machine learning-based approaches and machine learning-based approaches.

2.2.1 Non-machine learning-based LSR

The most straightforward and most widely used LSR methods are interpolation-based approaches. These methods estimate high-resolution load profiles by filling in values between consecutive low-resolution measurements, using only the surrounding data points. Common examples include the linear interpolation method [101, 102], which assumes a constant rate of change, or the cubic interpolation method [103, 104], which models gradual changes in slope between load measurements. Those methods are computationally efficient and easy to implement in practical applications. They can perform well when the scaling factor (defined as the ratio between the high-to low-resolution data) is low. When the scaling factor increases, their performance deteriorates dramatically because the underlying algorithms lack the complexity required to reconstruct the high-frequency variations present in the true load profiles. They are also highly sensitive to outliers and random noise, both of which are frequently present in practical load measurements.

Besides interpolation methods, signal processing techniques have also been widely applied to LSR. This is because energy load profiles are discretely sampled by smart meters, and thus inherently follow the principles of discrete-time signal processing. The general concept is to decompose load profiles into multiple components using a specific signal processing algorithm. Each component is then processed separately to enhance or preserve its characteristics within its respective band. Finally, the processed components are recombined to reconstruct the high-resolution load profile. For example, [105, 106] employs the Fourier transform to convert the energy load profiles from the time domain into the frequency domain. In the frequency domain, the components that contain rapid shape variations and peak value fluctuations are explicitly separable from the low-frequency components (i.e., contain long-term trends). By selectively enhancing those high-frequency components, the method restores fine details in load profiles. Analogously, [107, 108] applies the discrete wavelet transform to decompose the load series into multiscale detail coefficients. Similarly, [109] employs a synthesis filter-bank approach to partition the energy profiles into multiple sub-bands, while [110] adopts singular spectrum analysis by embedding the load series into a Hankel matrix. In each case, the fine-grained components are processed and enhanced separately to recover detailed information that is often lost in low-resolution measurements. Many of these signal processing techniques rely on the assumption that load profiles are stationary. That is, their statistical properties, such as mean and variance, remain constant over time. In practice, this assumption often fails because real-world energy load profiles exhibit significant non-stationary, driven by factors such as inherently unpredictable and volatile user behavior, weather variability, and operational events.

2.2.2 Machine learning-based LSR

Machine learning–based LSR methods are primarily built on some deep learning architectures. In typical training setups, the target outputs are high-resolution load

profiles, often obtained from publicly available datasets [111]. The corresponding low-resolution inputs are artificially generated from these high-resolution profiles by applying averaging operations, simulating the measurements obtained from low-resolution metering devices.

A common deep learning approach for LSR is to adopt CNNs as the network architecture. There are generally two ways to apply CNNs in this context. One is that since CNNs are typically designed for two-dimensional data, such as images, whereas energy load profiles are one-dimensional time-series, many LSR studies transform the one-dimensional load profiles into a 2-dimensional representation. For example, in [112], feature engineering is applied to construct a two-dimensional input: the first channel contains the sine-transformed of the load values, and the second channel contains the cosine-transformed of the load values. In [113], the study treats one year of time-series data as a single load profile and transforms it into a two-dimensional representation, where the horizontal axis corresponds to the days of a month (31 days) and the vertical axis corresponds to the months of the year (12 months). Other studies have adopted similar 2-dimensional transformation strategies for LSR, such as [114, 115].

Another way of applying CNNs is to design a one-dimensional variant of the convolutional layer specifically for time-series data, and to stack such layers to form a one-dimensional CNN. Compared with the approach of transforming the data for use with two-dimensional CNNs, this method is more parameter-efficient because it does not require handling an additional dimension, nor does it involve extra data preprocessing steps. As reported in recent studies [116-119], which show comparable or superior performance to two-dimensional approaches.

Another branch of deep learning–based LSR methods leverages Generative Adversarial Networks (GANs) [12, 120-125]. While conventional CNN-based LSR models can effectively reconstruct overall load profiles, their outputs often exhibit over-smoothed transitions at sharp edges or sudden load changes due to the inherent limitations of the training objective. That is, conventional deep learning–based LSR

models often adopt loss functions such as Mean Absolute Error (MAE) or Mean Squared Error (MSE) to update model parameters. While these loss functions are effective in minimizing overall prediction errors, they inherently encourage conservative reconstructions at extreme points or sharp transitions. In contrast, GAN-based approaches introduce an adversarial loss that encourages the model to better preserve such abrupt transitions, even at the expense of slightly higher errors in certain evaluation metrics (e.g., MAE or RMSE). These methods also retain a CNN-based architecture but employ a fundamentally different training paradigm. Usually, two CNN models are involved: a generator and a discriminator. During training, the generator is updated to produce increasingly realistic high-resolution reconstructions, while the discriminator is trained to distinguish between the generator's outputs and the actual high-resolution load profiles. This adversarial process encourages the generator to produce sharper transitions and more realistic high-frequency components than conventional loss-based training.

CHAPTER 3

Privacy-preserving Framework for Residential STLF

In this chapter, we address the challenges presented in Section 1.2 to build a privacy-preserving STLF framework for residential energy users. The framework enables users to collaboratively train an STLF model without exchanging their load data. The system works in two stages: (i) a federated user clustering method is developed to divide the users into multiple clusters based on load pattern similarity. Instead of aggregating the users' data to do clustering, the developed method determines the user clusters in a distributed manner. (ii) after the user clustering stage, a hierarchically federated STLF model training is developed, which applies federated learning principles to facilitate intra- and inter-user cluster model training. In the process, the individual users also do not expose their load data. Further, an asynchronous communication mechanism is designed and integrated into the framework, making it highly fault-tolerant and adaptable to the residential environment with communication uncertainties. Comprehensive experiments based on a real Australian load dataset are conducted to validate the system. The simulation results show that the proposed framework can effectively train the load forecasting model with a fast coverage rate and achieve up to 37.25% prediction accuracy improvement (in terms of the Mean Absolute Percentage Error metric) compared with the other benchmark methods.

3.1 Introduction

STLF [126] is a fundamental task in power and energy systems. Conventionally, load forecasting is performed at the bus level, i.e., to predict the aggregated load of a region comprising a large number of individual energy loads. In smart grids, deployment of

the AMI [127] has facilitated the automatic measurement of the power consumption of single energy users. As a result, STLF at the individual user level (i.e., to forecast the load of single users) has become an active topic in recent years, and it can support a variety of upper-level demand side management applications, such as the development of building/home energy management, customer-oriented energy recommendation systems, electricity retail pricing, and microgrid operational planning.

The STLF techniques reported in the literature can be broadly classified into two categories: statistical and learning-based methods. Some statistical methods, such as the Box and Jenkins transfer function [128] and linear regression methods [22], model the power load as a function of several influencing factors (e.g., temperature, social events, and weather conditions) and predict the future power load as the output of the function. Other statistical methods, such as the Kalman Filter [129, 130], ARIMA models [26, 28, 131], and the Exponential Smoothing method [132, 133], predict the future power load based on residential users' historical time series load data. Statistical-based STLF methods are computationally efficient and relatively easy to implement; however, they are limited in representing complex load patterns. Various learning-based methods have been developed for more effective STLF. These methods use load data to train certain machine learning models (e.g., support vector machines [134, 135], random forest regressors [136, 137], and LSTM recurrent networks [138, 139]) and use the trained models to predict the future load.

For the problem of single-user STLF, most of the existing work performs load forecasting based on the historical load data of the target user. These approaches can hardly be adapted to the situation when the target user has limited historical load data for training. Such a situation is common in modern cities (e.g., hotels, short-term accommodation, and newly moved-in households). Besides, solely using the historical load data of a single user for STLF could lead to the over-fitting problem [140], which degrades the load forecasting performance. One approach for tackling these issues is to practice the collaborative learning principle, i.e., to utilize multiple users' load data to "assist" the users with limited data for training the load forecasting model without

incurring the over-fitting issue. For example, in the authors' recent work [141], a transferrable model agnostic meta-learning-based STLF framework is developed. The method aggregates the data of multiple users to train a global model and then transfers the model to each target user for STLF purposes through a fine-tuning process. Although the framework has been proven effective, it is associated with a privacy-preserving issue. That is, load data would be considered as the user's private information. Simply collecting data from different users to train a model (as one way to train an LSTM in [139]) may not apply to users with high privacy-preserving concerns. This issue is more noticeable when the entity that performs load forecasting only possesses the data of a single customer it manages (e.g., the home energy management system developer of a specific customer).

Aiming to facilitate collaborative learning-based STLF while preserving the users' data privacy, Federated Learning (FL)-based STLF methods have been developed [142-147]. [142] develops an FL system for the single-user level STLF. In the system, each user trains a local load forecasting model based on its load data. Then, the users submit the parameters of their local models to a central node. The central node uses the aggregated parameters to determine the parameters of a global STLF model and sends its parameters to the users. The users then use the received parameters to update their local models further. This process is iteratively performed until a given termination criterion is met. In this way, different users can share spatial-temporal correlation (as represented by the model parameters) without sharing their load data. However, directly applying the FL process to STLF can lead to an issue: when the load patterns of the users are highly diverse, the spatial-temporal correlation of one user would have little reference to the others – in this situation, sharing model parameters among the users could delay the global model convergence, degrading the STLF performance.

To overcome the above issue, the recent literature ([144, 145] and the authors' work [143]) introduces a load clustering operation before performing the FL-based STLF process. With such a design, the load clustering operation can improve the system's STLF performance. It can ensure that load forecasting knowledge is only

shared among users exhibiting similar load patterns. By combining load clustering and FL-based STLF, the users are firstly divided into multiple clusters based on the similarity of their load profiles or information; then, for each cluster, an FL process is applied to enable users to train an STLF model collaboratively.

Despite the apparent reasonability of integrating load clustering into STLF, those work [143-145] remains three unsolved problems: (i) firstly, with the load clustering strategy, users' personal information needs to be centrally aggregated for clustering; this conflicts with the initiative of using FL, i.e., to avoid revealing the data of individual users. In the literature, two techniques have been developed to preserve the users' data privacy in load clustering [148, 149]. [148] develops an average consensus algorithm to perform clustering algorithms (i.e., *K*-means, fuzzy *C*-means clustering, and Gaussian mixture methods) in a distributed manner. This mechanism is fully decentralized and would not be suitable for FL-based STLF systems, since it does not utilize the aggregation node(s) that exist in FL systems, which would lead to a large number of communications among the users. [149] develops distributed *K*-means clustering techniques for load clustering. Despite its effectiveness in preserving the users' data privacy, the method does not include a mechanism to mitigate the negative impact that outlier load profiles would cause; in addition, it only performs clustering for load profiles rather than users (a user is characterized by a group of historical load profiles), and thus can hardly be directly applied to FL-based STLF systems. In FL-based STLF systems, the FL process needs to be performed in a group of users with similar load patterns rather than in a group of load profiles. (ii) Secondly, for the methods in [143-145], FL is performed separately in different user clusters, and there is no communication among clusters. This makes them not able to utilize the rich latent information among users sufficiently and would degrade the performance of the collaborative STLF scheme when the number of users in a cluster is small; (iii) thirdly, the existing FL-based STLF methods [142-147] are based on synchronous communication among the users. In each iteration of the FL process, the central node does not start the parameter update for the global model until it collects the local models'

parameters from all the users. Such a synchronous communication requirement makes the methods hard to adapt to the real-world residential environment. Compared with critical infrastructures that are usually equipped with dedicated and well-protected communication networks, the residential environment could be highly dynamic. For example, it could be communication delays, packet losses, and even loss of nodes (e.g., a user may leave the FL-based STLF process without notification). This indicates the necessity of more robust federated STLF system designs in the residential sector.

3.1.1 Contribution

To overcome the above-identified limitations in the existing work, this study proposes a new STLF framework that enables users to collaboratively train an STLF model with high performance in a privacy-preserving manner. The system performs STLF following two stages: firstly, a federated user clustering method is developed to divide the users into multiple clusters based on their load pattern similarity. After that, hierarchically federated learning principles are developed to train an STLF model for the users. The specific contributions that distinguish the proposed framework from the existing methods are 3-fold:

1. To develop a federated, privacy-preserving user clustering method. Instead of centrally aggregating the users' load data to practice a clustering procedure (as done in [143]), the developed framework performs load clustering in a distributed manner without aggregating the users' load data. Compared with the work in [149], the proposed method integrates a module that assigns users to different clusters based on the load profile clustering result, allowing the method to seamlessly connect to the FL-based STLF system; the method also integrates an outlier load profile identification and removal mechanism, which can contribute to enhanced load clustering performance.
2. To develop a hierarchically federated mechanism to train the STLF model. The mechanism applies the FL principle to facilitate STLF model training at intra-

and inter-user cluster levels. Such a unique design enables users with similar load patterns to intensively collaborate to train the forecasting model while utilizing the data features from all the users' load profiles. As a result, the STLF accuracy can be significantly improved. Moreover, the users' load profiles are not exposed during the training process. Together with the federated user clustering technique in (1), the system can sufficiently preserve the users' data privacy.

3. Unlike the synchronous communication pattern used in the existing FL-based STLF methods, an asynchronous communication mechanism is designed in the proposed framework, which makes it highly fault-tolerant and well-adapted to the residential scenario with communication uncertainties.

The rest of this chapter is organized as follows: Section 3.2 provides an overview of the proposed STLF framework. Sections 3.3 and 3.4 present the technical details of the federated user clustering method and the hierarchically federated STLF mechanism. Section 3.5 reports and analyzes the experiments. Section 3.6 discusses the conclusion.

3.2 Overview of the Proposed STLF Framework

The schematic of the proposed STLF framework is shown in Figure 3-1. The system works on a group of residential users, and the power consumption data from each user's smart meter is confidential. The system facilitates users to train an STLF model in a distributed manner collaboratively.

3.2.1 Federated User Clustering

As discussed in Section 3.1, collaborative training an STLF model suffers if users with highly diverse load patterns. While recent studies show that practicing a clustering operation to ensure FL training is performed among users with similar load patterns can improve the accuracy of the forecasting model [143, 144], they do not provide mechanisms to preserve load data privacy in the clustering process.

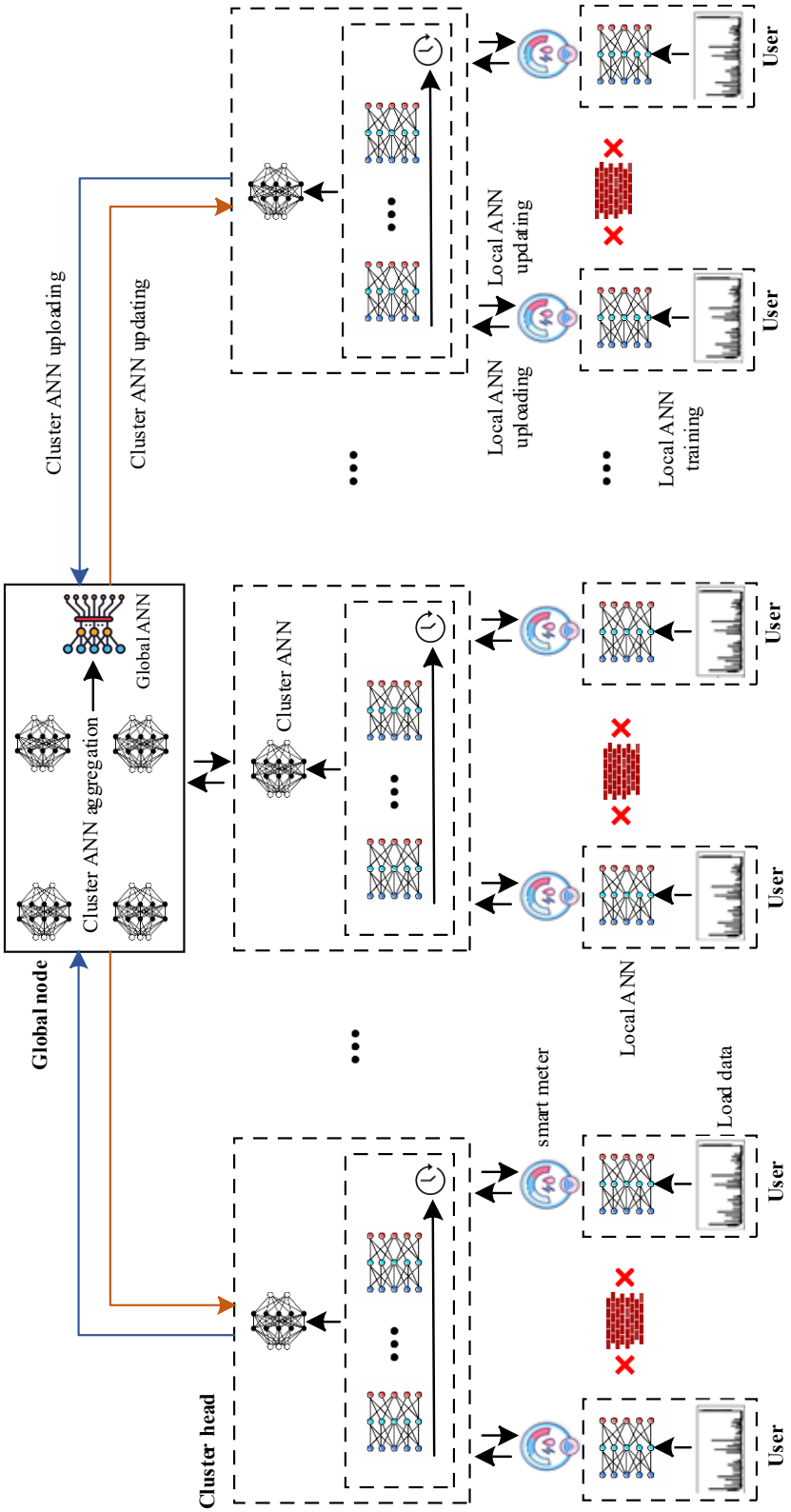


FIGURE 3-1. Schematic of the privacy-preserving and hierarchically federated STLF framework for residential energy users (the red crosses indicate no communication between the users).

The user clustering operation in the proposed STLF framework can sufficiently preserve the users' data privacy while effectively identifying their load pattern similarity. The method enables users to collaboratively perform a *K*-means load clustering algorithm in a distributed manner. Instead of submitting their private load data to a central node for clustering, the users communicate with each other to determine the centroids of the clusters without exposing their load data to a central node.

3.2.2 Hierarchically Federated STLF for Residential Users

After residential users are split into different clusters, a cluster head (CH) is set up in each cluster. Meantime, a global node (GN) is set up in the system. The CHs, GN, and each user initialize an Artificial Neural Network (ANN). The ANNs hosted by the GN, the CHs, and the users are named the global ANN, cluster ANNs, and local ANNs, respectively.

The GN, CHs, and users train ANNs following a hierarchically federated process. At the lower level, the CHs launch an FL process with the users belonging to the same cluster to train the cluster ANN for each cluster. The CHs and the GN perform another FL process at the upper level to train the global ANN. This hierarchical FL process is performed iteratively. When the iteration terminates, the global ANN is distributed to the users. Each user applies it for STLF based on its load data.

3.2.3 Communication Robustness and Data Privacy Preserving

Unlike conventional FL-based STLF methods built upon the synchronous communication pattern, where the server (this refers to the CH in the proposed framework) must wait for the arrival of the parameters of all the local ANNs to update the global model, the proposed framework uses an asynchronous communication pattern. That is, the CH immediately updates the cluster ANN when receiving the parameters of a local ANN. In this way, the framework can be well suited to users with different bandwidths and communication latency (this is the common case in the

residential sector); it can also tolerate the loss of communication messages from one or more users.

In summary, by integrating the federated user clustering and hierarchically federated STLF modules, the proposed framework can effectively facilitate collaborative STLF among users without triggering data privacy concerns. Further, the asynchronous FL mechanism makes the framework highly robust for the communication uncertainty of the residential environment. The technical details of the framework will be presented in Sections 3.3 and 3.4.

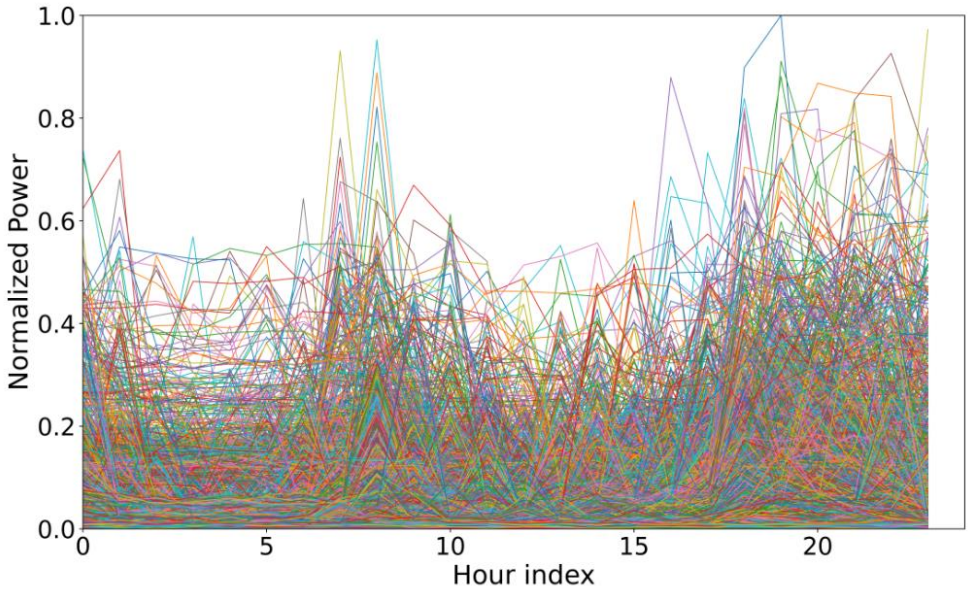
3.3 Federated *K*-means User Clustering

Residential power loads usually exhibit highly diverse patterns. As a demonstration, Figure 3-2 (a) shows the 24-hour normalized power load profiles of 100 residential users in the Greater Sydney area (data source: the Australian "Smart Grid, Smart City (SGSC)" customer trial dataset [150]) from June 01 to September 01, 2013. Directly performing FL on such load data could lead to unsatisfactory STLF performance, and a clustering operation could be helpful to group users with similar load patterns. Figure 3-2 (b) demonstrates the results generated from a load profile clustering operation. The profiles are grouped into 3 clusters, each of which is marked by a color.

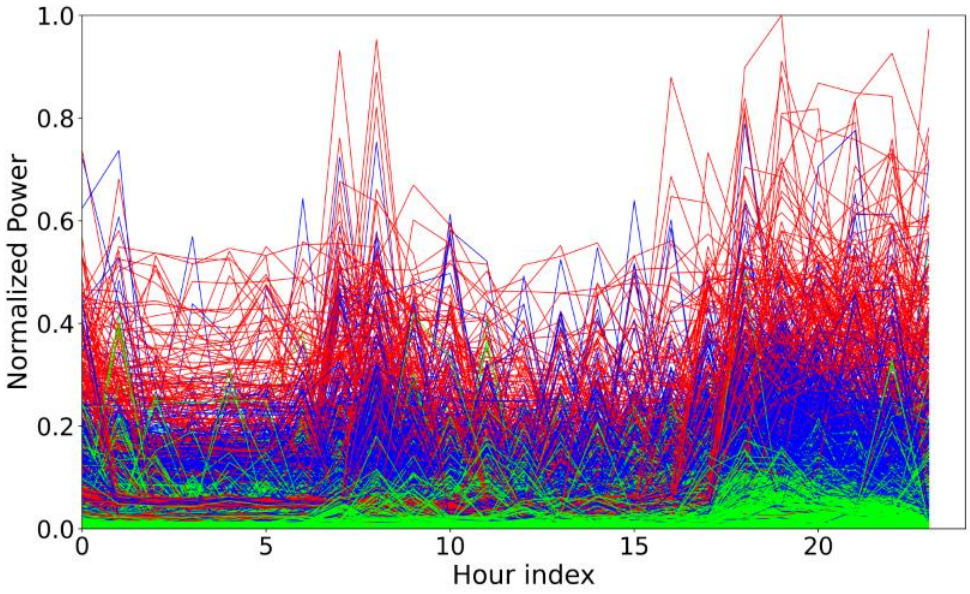
The user clustering is backboned by a federated clustering algorithm, which divides a group of data points into multiple clusters by minimizing the sum of the distances between each data point and the centroid of the cluster to which the data point is assigned:

$$F = \sum_{k=1}^{K} \sum_{z=1}^{X} \left\| x_z - \mu_k \right\|^2 \tag{3.1}$$

where K is a pre-specified parameter that denotes the number of clusters; X is the total number of data points; μ_k is the centroid of the kth cluster ($k = 1:K$); x_z denotes the value of the zth data point ($z = 1:X$). Compared with the conventional K-means clustering method, the federated user clustering method is trained in a distributed way,



(a)



(b)

FIGURE 3-2. Demonstrations of 24-hour normalized power load profiles of 100 users in the Greater Sydney area: (a) without load profile clustering; (b) with load profile clustering. In (a), each curve represents the load profile of a user. In (b), each color indicates a load cluster.

which does not need to aggregate the users' load data to a central system.

3.3.1 Representation of User's Load Data

Consider there are *M* users to be clustered; Each user's power load dataset can be expressed as a 1-dimensional time series vector:

$$\mathbf{P}_m = [P_{m,1}, ..., P_{m,Q_m}] \tag{3.2}$$

where *m* is the user index ($m = 1:M$); Q_m is the total number of load readings of the *m*th user; $P_{m,q}$ $(q = 1:Q_m)$ is the power load reading of the user in the *q*th hour (kW). For clustering purpose, $\mathbf{P}_m$ is transformed into a matrix form:

$$\mathbf{P}_m = \begin{bmatrix} P_{m,1,1} & \cdots & P_{m,1,T_m} \\ \vdots & \ddots & \vdots \\ P_{m,D_m,1} & \cdots & P_{m,D_m,T_m} \end{bmatrix} \tag{3.3}$$

In the matrix in Eq. (3.3), each row represents a load profile consisting of *T* load values recorded at a regular time interval (the length of one historical load profile), and D_m is the number of the historical load profiles of the *m*th user; The entry $P_{m,d,t}$ $(d = 1:D_m,\ t = 1:T_m)$ in the matrix denotes the *m*th user's power load at the *t*th time slot of the *d*th historical load profile. One historical load profile is with *T* dimensions. Based on this, the load data of a user in Eq. (3.3) can be rewritten as:

$$\mathbf{P}_m = \begin{bmatrix} \mathbf{P}_{m,1} \\ \mathbf{P}_{m,d} \\ \cdots \\ \mathbf{P}_{m,D_m} \end{bmatrix},\ m = 1:M \tag{3.4}$$

$$\begin{gathered} \mathbf{P}_{m,d} = [P_{m,d,1}, P_{m,d,2}, ..., P_{m,d,T_m}], \\ d = 1:D_m,\ m = 1:M \end{gathered} \tag{3.5}$$

3.3.2 Determination Process of Cluster Centroids

All *M* users' historical load profiles are used to determine the *K* clusters' centroids. Firstly, the GN initializes a centroid for each cluster. The centroids are denoted as

$\mathbf{C}_i = [\boldsymbol{\mu}_{i,1},...,\boldsymbol{\mu}_{i,K}]$, where $\boldsymbol{\mu}_{i,k}$ denotes the centroid of the *k*th ($k = 1:K$) cluster in the *i*th training iteration ($i = 1:I$, where I denotes the maximum number of training iterations for clustering). The centroids are then updated in an iterative manner coordinated by the GN. In each iteration, the GN randomly selects a set of users (denoted as $\boldsymbol{\Theta}_i$) and broadcasts the centroids $\mathbf{C}_i$ to them. After receiving the centroids, each user in $\boldsymbol{\Theta}_i$ calculates the Squared Euclidean Distance between each load profile and the cluster centroids, assigning each load profile to the cluster with the shortest distance:

$$\begin{aligned} \omega_{i,m,d} &= \arg\min_k^K \left\| \mathbf{P}_{m,d} - \boldsymbol{\mu}_{i,k} \right\|^2, \\ &\forall m \in \boldsymbol{\Theta}_i,\ d = 1:D_m \end{aligned} \tag{3.6}$$

where $\omega_{i,m,d}\pi$ is an auxiliary variable; it equals 1 if $\mathbf{P}_{m,d}$ is assigned to the *k*th cluster in the *i*th training iteration and it equals 0 otherwise. Then, each user calculates the sum of the distances between each load profile in its load dataset and the centroid assigned to it ($\mathbf{J}_{i,m}$):

$$\mathbf{J}_{i,m} = \sum_{d=1}^{D_m} \omega_{i,m,d} (\mathbf{P}_{m,d} - \boldsymbol{\mu}_{i,k}),\ \forall m \in \boldsymbol{\Theta}_i,\ k = 1:K \tag{3.7}$$

After applying Eq. (3.7), each users sends $\mathbf{J}_{i,m}$ and a vector $\mathbf{V}_{i,m} = [V_{i,m,1},...,V_{i,m,K}]$ to the GN. $\mathbf{V}_{i,m}$ records the numbers of the load profiles in the user's data that have been assigned to each cluster. The GN aggregates $\mathbf{J}_{i,m}$ and $\mathbf{V}_{i,m}$ from $\boldsymbol{\Theta}_i$:

$$\mathbf{J}_i = \sum \mathbf{J}_{i,m},\ \forall m \in \boldsymbol{\Theta}_i \tag{3.8}$$

$$\mathbf{V}_i = \sum \mathbf{V}_{i,m},\ \forall m \in \boldsymbol{\Theta}_i \tag{3.9}$$

The GN updates the centroids for the successive training iterations such as:

$$\mathbf{C}_{i+1} = \mathbf{C}_i + \alpha \cdot \frac{\mathbf{J}_i}{\mathbf{V}_i} \tag{3.10}$$

In Eqs. (3.8-3.10), the vector operations are performed elementwise. In Eq. (3.10), α is a pre-specified learning rate for the proposed clustering method. The above steps proceed until one of the two criteria is satisfied: (i) the pre-set maximum iteration number is reached; or (ii) the distances updated between all the load profiles and the centroids to which the load profiles belong is smaller than a pre-specified threshold ρ. It can be seen that the actual load data of the users is not transmitted in the cluster centroid determination process; instead, only $\mathbf{J}_{i,m}$ and $\mathbf{V}_{i,m}$ are transmitted from users. Since this information items are just about the distance between the load profiles and the centroid and the cluster assignment information of the load profiles, it would be difficult for an attacker to infer the user's load data or meaningful load patterns (e.g., peak power consumption and peak-to-value ratio) from this information. In this sense, the method can sufficiently preserve the user's data privacy.

3.3.3 Identification and Removal of Outlier Load Profiles

It has been recognized that the outlier data points would negatively affect the performance of a clustering algorithm [151]. This also applies to the distributed user clustering method in this study, as the outlier load profiles would significantly affect the determination of the cluster centroids. Based on this, we design a mechanism to reduce the load profile outliers, which works as follows:

After determining the cluster centroids following the method in Section 3.3.2, for each user, the distance between each load profile in the user's historical load dataset and the centroids are calculated:

$$s_{k,m,d} = \left\| \mathbf{P}_{m,d} - \boldsymbol{\mu}_k \right\|,\ d = 1:D_m,\ m = 1:M,\ k = 1:K \tag{3.11}$$

Then, the load profile is assigned to the centroid with the minimum value of $s_{k,m,d}$ (denoted this shortest distance as $s^*_{m,d}$). The load profiles that satisfy the condition of $s^*_{m,d} > s^p_m$ are considered as outliers and removed from the user's dataset, where s^p_m

is a pre-set percentile threshold. Afterwards, the cluster centroids are updated by applying the steps in Section 3.3.2 one more time.

3.3.4 Assigning Clusters to Users

After the cluster centroids are determined, each user needs to be classified into a cluster to launch the hierarchically federated STLF process. To do this, each user computes its average daily power load profile ($\hat{\mathbf{P}}_m$):

$$\hat{\mathbf{P}}_m = \frac{1}{D_m}\sum_{d=1}^{D_m}\mathbf{P}_{m,d},\ m=1:M \tag{3.12}$$

Then, each user assigns itself to one cluster based on $\hat{\mathbf{P}}_m$, which applies:

$$\hat{\omega}_m = \arg\min_k^K \left\|\hat{\mathbf{P}}_m - \boldsymbol{\mu}_k\right\|^2,\ m=1:M \tag{3.13}$$

where $\hat{\omega}_m$ is an auxiliary variable that either equals 1 (if it belongs to the cluster) or 0. Algorithms 3.1, 3.2 and 3.3 describe the overall training procedures of the developed

ALGORITHM 3.1 Procedures of Federated *K*-means User Clustering

Inputs: number of clusters K; maximum iteration number I; clustering learning rate α; threshold ρ; the users' load datasets ($\mathbf{P}_m, m = 1:M$).
Output: cluster centroids $\mathbf{C}$.

```
Randomly initialize the cluster centroid C_0;
For i = 1:I, Do
    Randomly select a set of users Θ_i;
    For users in Θ_i in Parallel, Do
        (J_{m,i}, V_{m,i})→Algorithm 3.2 (C_i, P_m);
    End For
    Update J_i and V_i following Eqs. (3.8) and (3.9);
    Update cluster centroids to be C_{i+1} following Eq. (3.10);
End For
Perform Algorithm 3.3 to remove outlier load profiles;
Repeat steps 2-9 to update C.
Output C.
```

ALGORITHM 3.2 Sub-Routine Performed by Each User

Inputs: cluster centroids $\mathbf{C}_i$; the users' load datasets ($\mathbf{P}_m$, $m=1:M$).
Output: distance matrix $\mathbf{J}_{m,i}$; cluster information $\mathbf{V}_{m,i}$.

Initialize elements in $\mathbf{V}_{m,i}$ to be 0;
For each load profile $\mathbf{P}_{m,d}$ ($d=1:D_m$), **Do**
    Assign $\mathbf{P}_{m,d}$ to a cluster following Eq. (3.6);
End For
Compute distance $\mathbf{J}_{m,i}$ following Eq. (3.7);
Update cluster information $\mathbf{V}_{m,i}$;
Output $\mathbf{J}_{m,i}$, $\mathbf{V}_{m,i}$.

ALGORITHM 3.3 Outlier Removal Performed by Each User

Inputs: cluster centroids $\mathbf{C}$; the users' load datasets ($\mathbf{P}_m$, $m=1:M$); and percentile threshold s_m^p.
Output: the users' load datasets with the removal of outlier load profiles.

For each user $m=1:M$, **Do**
    For each load profile $\mathbf{P}_{m,d}$ ($d=1:D_m$), **Do**
        Calculate the distances between $\mathbf{P}_{m,d}$ and the centroids following Eq. (3.11);
        Determine $s_{m,d}^*$ for $\mathbf{P}_{m,d}$;
        If $s_{m,d}^* > s_m^p$, **Do**
            Remove $\mathbf{P}_{m,d}$ from $\mathbf{P}_m$;
        End If
    End For
End For
Output $\mathbf{P}_m$, $m=1:M$.

federated user clustering method. Algorithm 3.1 is executed by the GN, which iteratively communicates with the users to execute Algorithm 3.2 and 3.3 to perform the distributed clustering process.

3.4 Hierarchically Federated STLF Mechanism

After the users are divided into multiple clusters by applying the privacy-preserving user clustering method in Section 3.3, a two-layer federated learning mechanism is applied to enable the users to train the ANN model for STLF. The mechanism applies

the FL principles at the intra- and inter-cluster levels. In this section, we first introduce the basic format of the input and output data of the ANN models for STLF purposes; then, the proposed hierarchically federated STLF mechanism is presented.

3.4.1 Inputs and Outputs of the ANNs

For each user, the sliding window method generates the input and output load data pairs of the ANN. Specifically, the load readings over L^p time slots are used to generate the predicted load values of the next L^f future time slots. Specifically, based on a user's historical power load data represented in Eq. (3.2), the input and output sets of the ANN (denoted as $\mathbf{X}_m$ and $\mathbf{Y}_m$, respectively) can be represented as:

$$\mathbf{X}_m = \begin{bmatrix} x_1 & \cdots & x_{L^p} \\ \vdots & \ddots & \vdots \\ x_J & \cdots & x_{J+L^p-1} \end{bmatrix}, \ \mathbf{Y}_m = \begin{bmatrix} x_{L^p+1} & \cdots & x_{L^p+L^f} \\ \vdots & \ddots & \vdots \\ x_{J+L^p} & \cdots & x_{Q_m} \end{bmatrix} \tag{3.14}$$

where $J = Q_m + 1 - L^p$. Each pair of rows in $\mathbf{X}_m$ and $\mathbf{Y}_m$ is used as a set of ANN input and output. The parameters of the ANN are trained in an iterative manner. In each iteration, the parameters are updated based on a certain number of input and output pairs extracted from $\mathbf{X}_m$ and $\mathbf{Y}_m$ (called a "batch"). The number of such input and output pairs in each batch is called batch size, which is a hyperparameter that needs to be specified before training.

3.4.2 Hierarchically Federated STLF Model Training

For each cluster, a CH is set up. Besides, a GN is also set up in the system during the user clustering stage. There is an ANN in each user, each CH, and the GN. The ANN in the GN is called the global ANN; the ANNs in the CHs are called the cluster ANNs; and the ones on the user side are called the local ANNs. The hierarchically federated STLF process is then performed among the users, CHs, and GN. The CHs act as the intermediate agents between the users and the GN. Each CH communicates with the

users in the cluster to update the cluster ANN of the users through an intra-cluster federated ANN model training; the CHs communicate with the GN to update the global ANN through an inter-cluster federated ANN model training.

3.4.2.1 Intra-cluster Federated ANN Model Training

Firstly, the GN initializes the parameters of the global ANN (denoted as $\boldsymbol{\theta}_0^g$, where the subscript "0" indicates the initialization ANN parameters). $\boldsymbol{\theta}_0^g$ is then distributed to the CHs to be used as the initial models of the cluster ANNs, (i.e., $\boldsymbol{\theta}_{0,k}^c = \boldsymbol{\theta}_0^g$). In each iteration, each CH randomly selects a set of users in the cluster (denoted as $\boldsymbol{\Phi}_{n,k}$, $n = 1:N$, where N are the maximum communication iteration index of the hierarchically FL mechanism) and sends the cluster ANN's parameters $\boldsymbol{\theta}_{n,k}^c$ to the users that belong to the cluster. In each user cluster, after receiving the model parameters from the CH, each user applies the received parameters to update its local ANN by performing the Stochastic Gradient Descent (SGD) method [152] to its own load dataset:

$$\begin{aligned} &\boldsymbol{\theta}_{n,m,w+1}^l = \boldsymbol{\theta}_{n,m,w}^l - \gamma \nabla_{\boldsymbol{\theta}_{n,m,w}^l} \ell_{\boldsymbol{\theta}_{n,m,w}^l}(\boldsymbol{\theta}_{n,m,w}^l; \mathbf{x}_m, \mathbf{y}_m), \\ &\forall m \in \boldsymbol{\Phi}_{n,k}, w = 1:W, \forall \mathbf{x}_m \in \mathbf{X}_m, \forall \mathbf{y}_m \in \mathbf{Y}_m \end{aligned} \tag{3.15}$$

where $\boldsymbol{\theta}_{n,m,0}^l = \boldsymbol{\theta}_{n,k}^c$, and the subscript "0" denotes the initial parameters of the local ANN in the update process. γ is the learning rate; w and W denote the training iteration index and the maximum training iteration number of the local ANN, respectively; $\mathbf{x}_m$ and $\mathbf{y}_m$ are the one batch randomly extracted from the user's local data $\mathbf{X}_m$ and $\mathbf{Y}_m$, respectively; $\ell_{\boldsymbol{\theta}_{n,m,w}^l}(\cdot)$ denotes the loss function of the local ANN, which is expressed as the Mean Squared Error (MSE) calculated from the actual and predicted load values:

$$\ell_{\boldsymbol{\theta}_{n,m,w}^l}(\boldsymbol{\theta}_{n,m,w}^l; \mathbf{x}_m, \mathbf{y}_m) = \sum_{\forall \mathbf{x}_m, \forall \mathbf{y}_m \in \mathbf{X}_m, \mathbf{Y}_m} \left\| f_{\boldsymbol{\theta}_{n,m,w}^l}(\mathbf{x}_m) - \mathbf{y}_m \right\|_2^2 \tag{3.16}$$

After W times of local training following Eq. (3.15), each user sends the local ANN's parameters (denoted as $\boldsymbol{\theta}_{n,m}^{l}$) back to the CH. The CH then updates the cluster ANN. Unlike the existing FL-based STLF methods [142-147] that perform FL in a synchronized communication manner, i.e., the CH aggregates the local ANN parameters of all the users in $\boldsymbol{\Phi}_{n,k}$ to update the cluster ANN, in the proposed framework, such an update is performed in an asynchronous communication manner – that is, once the CH has received the local ANN from a user, it immediately updates the cluster ANN:

$$\boldsymbol{\theta}_{n,k}^{c} = (1-\beta)\boldsymbol{\theta}_{n,k}^{c} + \beta\boldsymbol{\theta}_{n,m}^{l}, \forall m = \boldsymbol{\Phi}_{n,k}, n = 1:N \tag{3.17}$$

where β is a pre-specific weighting factor that balances the roles of the existing cluster ANN and the local ANN received from a user in the update of the cluster ANN. After a pre-specified time, the cluster ANN update process terminates, and the local ANN parameters that have not been sent from one or multiple users to the CH (e.g., due to communication latency or congestion) are discarded. The parameters of the updated cluster ANN are then sent to the users in the cluster to further train the local ANNs.

3.4.2.2 Inter-cluster Federated ANN Model Training

The above process of model training between the CH and the users is iteratively performed for R iterations. After every R-iteration, each CH sends the parameters of the cluster ANNs to the GN. The GN then updates the global ANN:

$$\boldsymbol{\theta}_{n}^{g} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{\theta}_{n,k}^{c}, n = 1, R, 2R, ..., N \tag{3.18}$$

The global ANN's parameters $\boldsymbol{\theta}_{n}^{g}$ are then evaluated on a validation load dataset subjected to the MSE metric:

$$\ell_{\boldsymbol{\theta}_{n}^{g}}(\boldsymbol{\theta}_{n}^{g}; \mathbf{X}^{v}, \mathbf{Y}^{v}) = \left\| f_{\boldsymbol{\theta}_{n}^{g}}(\mathbf{X}^{v}) - \mathbf{Y}^{v} \right\|_{2}^{2}, n = 1, R, 2R, ..., N \tag{3.19}$$

where $\mathbf{X}^{v}$ and $\mathbf{Y}^{v}$ denote the input and output of the global ANN based on the

validation dataset, respectively. Afterward, $\boldsymbol{\theta}_n^g$ is distributed to the CHs to train the cluster ANNs in the next iteration, i.e., $\boldsymbol{\theta}_{n+1,k}^c = \boldsymbol{\theta}_n^g,\ (k = 1:K)$.

The above whole hierarchically federated model training process is iteratively performed until a pre-specified maximum iteration number is reached. The global ANN with the lowest MSE value on the validation load dataset (following Eq. (3.19)) is selected as the final STLF model and is distributed to all the users for load forecasting.

3.5 Experiments

Experiments are conducted to verify the proposed STLF framework. All programs are implemented using Python and are executed on a computer with an Intel i7-12700F processor and a 24-GB GeForce GTX 3090Ti GPU.

3.5.1 Simulation Setup

In this study, we use the Australian "Smart Grid, Smart City" (SGSC) customer trial dataset [150] to evaluate the proposed framework. The dataset contains the power consumption data of over 10,000 residents in New South Wales between 2010 to 2014. The AMI collects the data with a 30-minute resolution. In this simulation, we convert the time resolution of the data to 1 hour by taking the average value of two neighboring data readings. All the load data values are normalized between 0 and 1 for model training purposes:

$$\hat{P} = \frac{P - P^{\min}}{P^{\max} - P^{\min}} \tag{3.20}$$

where $\hat{P}$ and P represent the normalized and actual load values, respectively; $P^{\min}$ and $P^{\max}$ denote the minimum and maximum load values in the user's historical load dataset, respectively.

In the federated user clustering stage, the weekday load profiles of the users (from Monday to Friday) are used. The load profile is the daily power consumption with 24

time slots. The clustering learning rate α is set to be 1; I is set to be 50; the early termination threshold value ρ for clustering is set to be 0.0001; the outlier percentile threshold s_m^p is set to be 80%. While the proposed STLF framework is compatible with different kinds of ANN models, without loss of generality, in this study, a basic Fully Connected Network (FCN) with three hidden layers is used as the ANN model to validate the framework. In the FCN, each hidden layer contains 128 neurons; the ANN learning rate γ is set to be 0.01; Following [139] and [141], the sliding window hyper-parameters L^p and L^f are set to be 12 and 1; the maximum local training W is set to be 5; the maximum number of communication iterations (N) for STLF training is set to be 300. The batch size is set to be 32. In each communication iteration, each CH randomly samples 10% of the users in its cluster to form $\mathbf{\Phi}_{n,k}$ for training; The Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are used as the evaluation metrics of load forecasting performance:

$$\text{RMSE}=\sqrt{\frac{1}{W^{\text{total}}}\sum_{x=1}^{W^{\text{total}}}(P_x-P_x^{\text{pred}})^2} \tag{3.21}$$

$$\text{MAPE}=\frac{1}{W^{\text{total}}}\sum_{x=1}^{W^{\text{total}}}\left|\frac{P_x-P_x^{\text{pred}}}{P_x}\right|\times 100\% \tag{3.22}$$

where P_x and P_x^{pred} denote the xth pair of actual and predicted load values. We compare the proposed STLF framework with three benchmark methods:

(1) *Individual Learning (IL)-based method*. In this method, each user only applies its own historical load data to train an ANN and uses it for load forecasting. This represents the simplest way of STLF.

(2) *Federated Averaging (FedAvg)-based method* [142]. This method uses the basic FL mechanism to do STLF. A GN is set up to host a global ANN model; the parameters of the global ANN model are sent to individual users, which use the parameters to train their local ANN models. The updated parameters of the local ANN models are sent back to the GN, and the latter uses the weighted average on the collected parameters as the

global ANN model's parameters. This method is based on the synchronized communication mechanism – that is, the GN does not update the global ANN model until it has received the local model from all the users.

(3) *Asynchronous Federated Learning (AFL)-based method.* This benchmark method integrates an asynchronous communication mechanism into the basic FL process. When updating the global ANN model, instead of calculating the weighted average value of all the parameters of local ANN models, the global ANN is immediately updated once the GN has received the local ANN model parameters from a user. Such an asynchronous model update process can be defined as:

$$\boldsymbol{\theta}_n^g = (1-\beta)\boldsymbol{\theta}_n^g + \beta\boldsymbol{\theta}_{n,m}^l,\ \forall m \in \boldsymbol{\Phi}_n, n = 1:N \tag{3.23}$$

where $\boldsymbol{\theta}_n^g$ and $\boldsymbol{\theta}_{n,m}^l$ are the global and local ANN model at *n*th communication iteration, respectively; $\boldsymbol{\Phi}_n$ is the sampled users at *n*th communication iteration.

(4) *FedAvg-based method with user clustering (FedAvg-C)* [143]. In this method, the users are divided into different clusters. The FedAvg-based method is then applied to each cluster separately, in which the CH communicates with the users in the cluster by applying a FedAvg process to train the cluster ANN model. There is no communication among the user clusters.

3.5.2 Validation of User Clustering

In the first part of the simulation, we validate the distributed user clustering method presented in Section 3.3. We compare the method with two other methods that can be used in load clustering: (i) the conventional *K*-means clustering method; and (ii) the federated gradient sharing (GS) based load clustering method reported in [149].

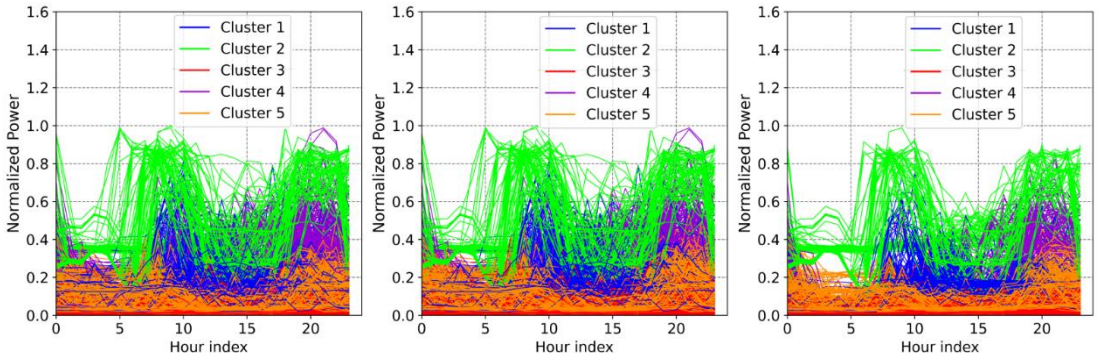
A total of 6,500 load profiles from 100 users are used for the validation. Figure 3-3 shows the comparison of the centroids of the load profiles under the 3 methods with the setting of five clusters (i.e., *K*=5). Figure 3-3 shows that all the 3 methods produce a similar load profile clustering result. Since the conventional *K*-means clustering method

is a centralized method, the results in Figure 3-3 validate the effectiveness of the distributed load profile clustering methods (i.e., the federated *K*-means-based method in this study and the federated GS method in [149]). These distributed methods can effectively divide the load profiles into different clusters as the centralized method without exposing the actual load data of the users.

Figure 3-4 shows the clustering results for 6,500 load profiles of the 100 users in detail, in which each color represents a cluster. Figure 3-4 (c) shows that with the federated *K*-means-based method, some outlier load profiles (e.g., the purple curves in the upper right corner and the green curves between hour index 10 and 15 of Figure 3-4 (a) and (b)) are removed by the outlier load profile identification and removal mechanism in Section 3.3.3. As a simple illustration, Figure 3-5 shows the load profiles of a user, in which some profiles are identified as outliers (red curves in the figure) and are removed in the clustering process. The conventional *K*-means and the GS- based methods are applied to load profile clustering; in contrast, the proposed federated *K*-means-based method performs an additional step to assign each user to a cluster based on the load profile clustering result (see Section 3.3.3). Figure 3-6 shows the cluster assignment results for 100 users.

(a) Centralized *K*-means method (b) Federated-GS method (c) Proposed federated *K*-means method

FIGURE 3-3. Comparison of the cluster centroids under different power load clustering methods.



(a) Centralized *K*-means method (b) Federated-GS method (c) Proposed federated *K*-means method

FIGURE 3-4. Clustering result for the normalized load profiles of the 100 users.

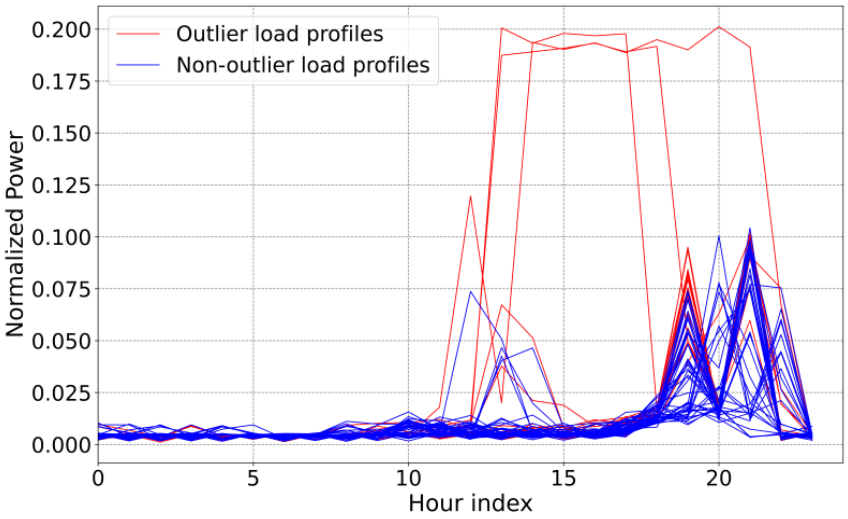


FIGURE 3-5. Demonstration of outlier and non-outliner load profiles of a user.

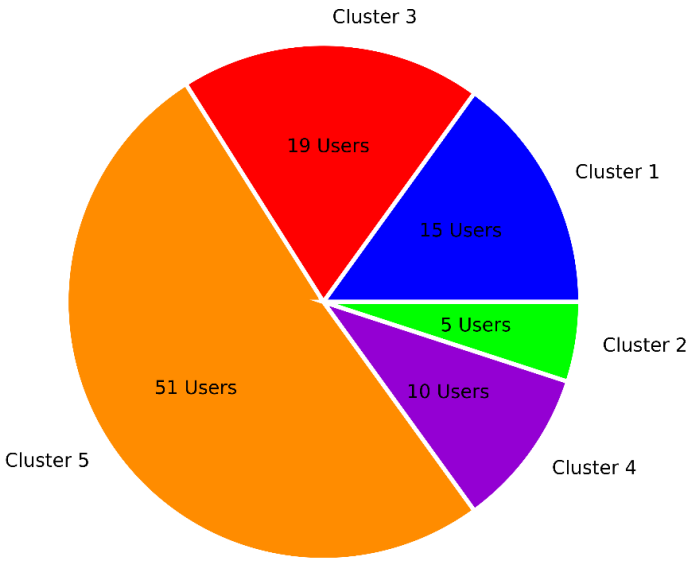


FIGURE 3-6. Numbers of users in different clusters.

3.5.3 Comparison of STLF Results

We validate the effectiveness of the proposed federated user clustering mechanism. For this purpose, we run the proposed framework under different settings of the user cluster number, i.e., K=2, 3, and 4, and compare the results with the FedAvg- and AFL-based methods. We randomly select 7-day and 21-day data from the dataset, respectively, to train the models. Figure 3-7 shows the average MSE values on the validation dataset over 5 independent trials in both the 7-day and 21-day training data cases. The former is designed to simulate a commonly seen real-world scenario where the training load data is highly limited and insufficient for the users. The result shows that clearly, the proposed framework outperforms the FedAvg- and AFL-based methods. When the users are divided into multiple clusters based on their load pattern similarity, the proposed framework can achieve fast model training. For example, in both 7-day and 21-day training data cases, the MSE value on the validation dataset generated by the proposed framework with K=3 after 50 communication iterations is lower than that generated by the FedAvg method after 300 communication iterations.

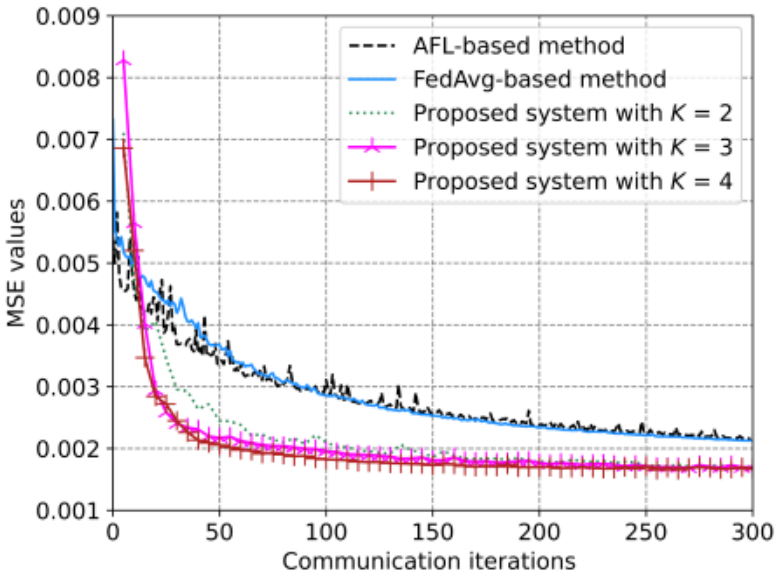
Table 3-1 reports the STLF results generated by the different methods. The FL-based methods (i.e., the proposed system and the AFL-based, FedAvg-based, and FedAvg-C methods) are trained with 300 communication iterations. Among those methods, the FedAvg-based method shows unsatisfactory STLF performance. Without integrating user clustering, the AFL-based method does not perform well either. In the 21-day training data case, these two methods perform even worse than the IL-based method (reflected in the higher MAPE and RMSE values). The reason for their unsatisfactory load forecasting performance could be that these two methods discriminately perform collaborative model training for all the users; as a result, the users with highly diverse load patterns would make the global FCN model hard to converge (as shown in Figure 3-7), which subsequently imposes a negative impact on the STLF accuracy. In contrast, with the introduction of the user clustering mechanism, the proposed system ensures collaborative model training is performed among users

with similar load patterns. This leads to a significant improvement in the STLF results. Table 3-1 also shows that the FedAvg-C method has worse STLF performance than the proposed system, especially in the 7-day training data case. The results indicate that, on the one hand, the proposed system can facilitate users with similar load patterns to intensively collaborate to train STLF models (i.e., the cluster ANNs); on the other hand, it can also take advantage of the rich features in the different cluster ANNs to train a model with enhanced STLF performance (i.e., the global ANN).

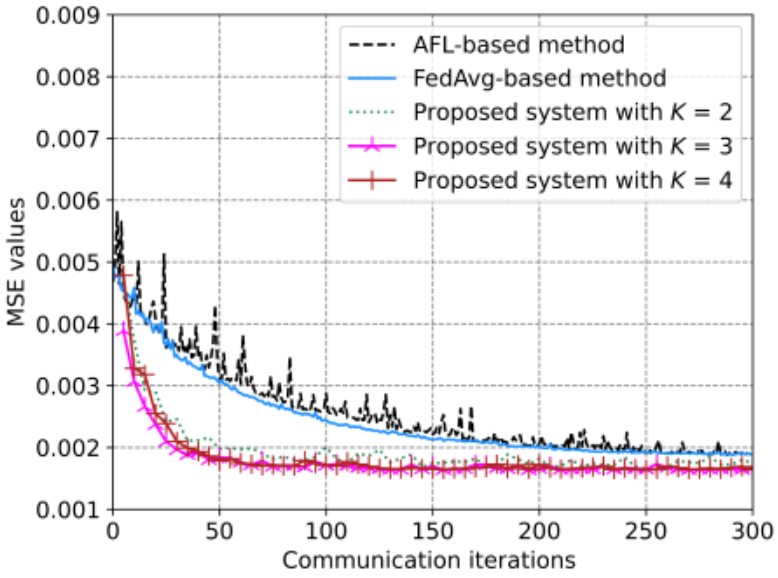
TABLE 3-1. Comparison of STLF performance among different methods

STLF methods	7-day training data						21-day training data					
	Best		Average		Worst		Best		Average		Worst	
	MAPE	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*
AFL-based method	109.73	0.160	121.34	0.164	142.15	0.169	121.28	0.156	124.62	0.158	127.88	0.160
Proposed system with K=2	85.26	0.150	93.16	0.150	**102.43**	**0.151**	75.45	0.148	91.45	0.149	110.06	0.152
Proposed system with K=3	86.26	0.149	93.57	0.150	109.20	0.151	**70.28**	**0.147**	**73.30**	**0.147**	**76.40**	**0.148**
Proposed system with K=4	**80.40**	**0.148**	**83.79**	**0.149**	107.60	0.153	82.45	0.148	82.51	0.148	109.89	0.151
FedAvg-C method with K=2	116.72	0.160	126.89	0.164	139.11	0.169	106.57	0.153	117.03	0.156	122.39	0.158
FedAvg-C method with K=3	124.17	0.161	128.86	0.165	134.44	0.169	114.08	0.153	114.40	0.156	115.24	0.159
FedAvg-based method	107.13	0.160	119.48	0.164	122.37	0.166	116.61	0.156	137.83	0.160	156.05	0.164
IL-based method	101.87	0.162	109.04	0.167	107.36	0.169	107.07	0.156	103.74	0.157	105.92	0.158

*For each method, the “best”, “average”, and “worst” results are generated from 5 independent trials; the “best” and “worst” results are based on RMSE metric.



(a)



(b)

FIGURE 3-7. Comparison of the convergence process of model training under the different FL-based STLF methods: (a) a 7-day training data case; (b) a 21-day training data case.

Figure 3-8 shows the method that achieves the best STLF accuracy across all 100 users in terms of MAPE and RMSE metrics. As mentioned before, the IL-based method outperforms the FedAvg- and AFL-based methods in performing STLF for most users. With K=3, the proposed system achieves the best MAPE values in 72 and 89 users and the best RMSE values in 79 and 56 users in the 7- and 21-day training data cases,

respectively.

Figure 3-9 shows two detailed examples of the 120-hour forecasted load profiles generated by the different methods for two users. The actual load profiles of the users are also plotted as solid black lines. In the 7-day training data case Figure 3-9 (a), for each user, there is a relatively large deviation between the actual and forecasted load profiles generated by the IL-based method. This indicates that the load forecasting model cannot be well trained by solely using the 7-day training data of the user. By enabling the users to collaboratively train the models following the proposed privacy-preserving user clustering and the hierarchically federated learning principles, the proposed system can well capture the features of the intra-day load variation, as shown in Figure 3-9, the forecasted load profiles generated by the proposed system with K=3 can well follow the sudden transitions in the actual load profiles. In the 21-day training data case (Figure 3-9 (b)), Since each user has more historical data, the STLF performances of all the methods have improved compared with the 7-day training data case, while the proposed system still performs best. Residential users' power consumption patterns could vary a lot in different seasons (as illustrated in Figure 3-10). We further compare the STLF performance for 100 users under the different methods in different seasons in Australia in 2013, based on the data provided in the SGSC dataset. For each season, the load data in the first 14 days of the season (i.e., September 1st to 14th for Spring; December 1st to 14th for Summer; March 1st to 14th for Autumn; and June 1st to 14th for Winter) is used for training the ANN models, and the load data in the next 14-day period is used for testing. The average results are reported in Table 3-2.

For each season, the proposed framework can achieve good generalizability on STLF performance. From the table, it can be seen that the STLF performance of the proposed framework outperforms the other methods in all 4 seasons. This is consistent with the results in Table 3-1 and it demonstrates how the proposed framework can well facilitate the different users to achieve effective collaborative STLF model training.

TABLE 3-2. Comparison of STLF under different seasons for 100 users

STLF methods	Spring (Sep.-Nov)		Summer (Dec.-Feb)		Fall (Mar.-May)		Winter (Jun.-Aug)	
	MAPE	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*
AFL-based method	76.95	0.139	68.96	0.154	75.01	0.165	77.39	0.194
Proposed system with K=2	**64.51**	**0.135**	**52.84**	**0.149**	**65.37**	**0.156**	**59.73**	**0.191**
FedAvg-C method with K=2	81.04	0.142	64.83	0.154	73.36	0.164	76.30	0.193
FedAvg-based method	73.40	0.137	62.16	0.153	88.12	0.170	86.06	0.198
IL-based method	87.05	0.152	80.40	0.173	90.54	0.183	95.18	0.203

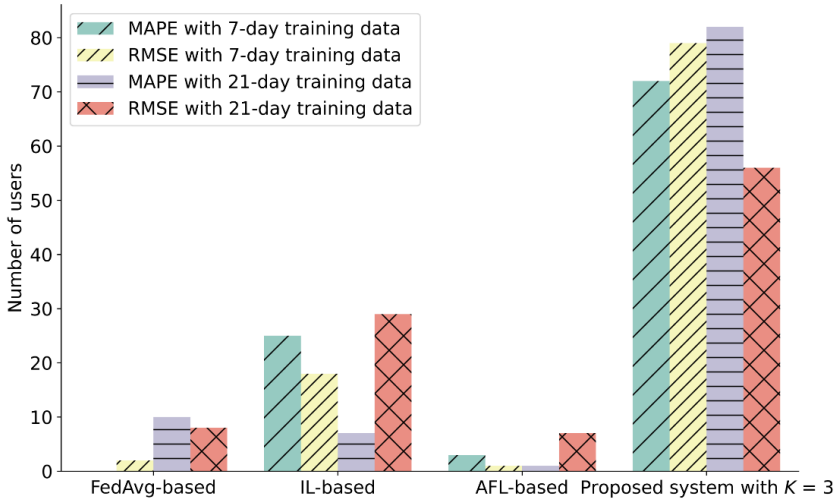
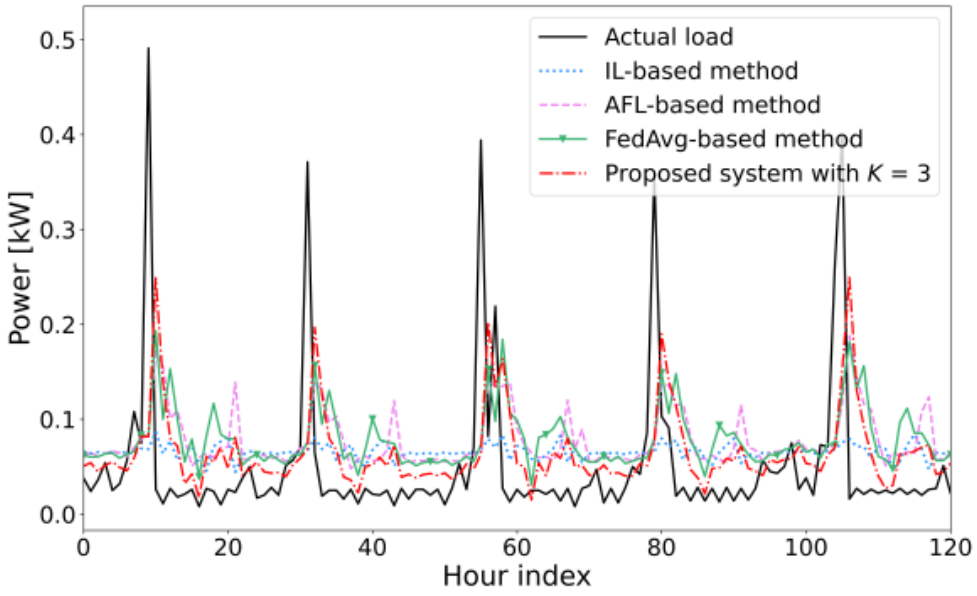
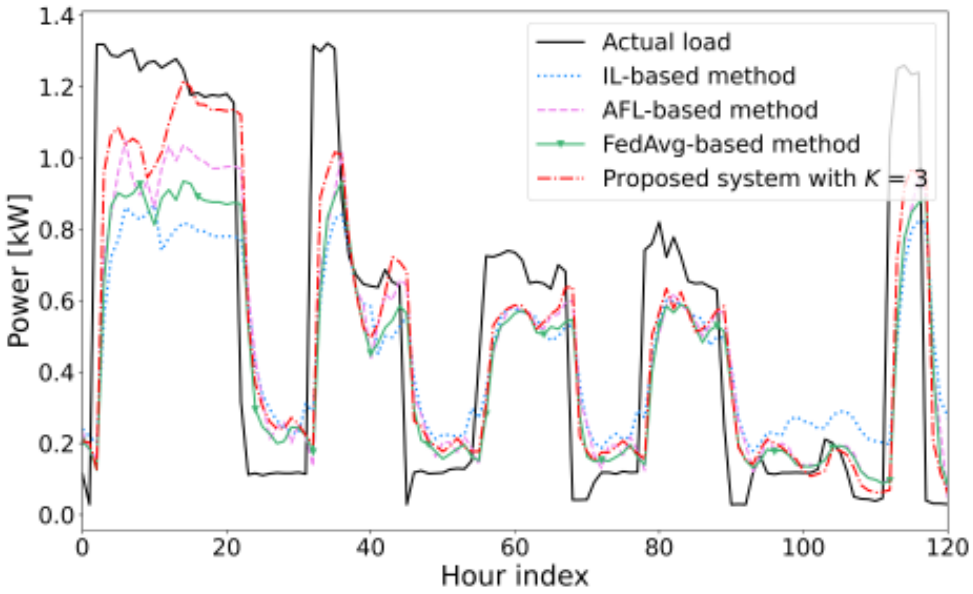


FIGURE 3-8. Statistics of the best STLF method among the 100 users.



(a)



(b)

FIGURE 3-9. 120-hour STLF result comparison of the different methods on two users: (a) a 7-day training data case; (b) a 21-day training data case.

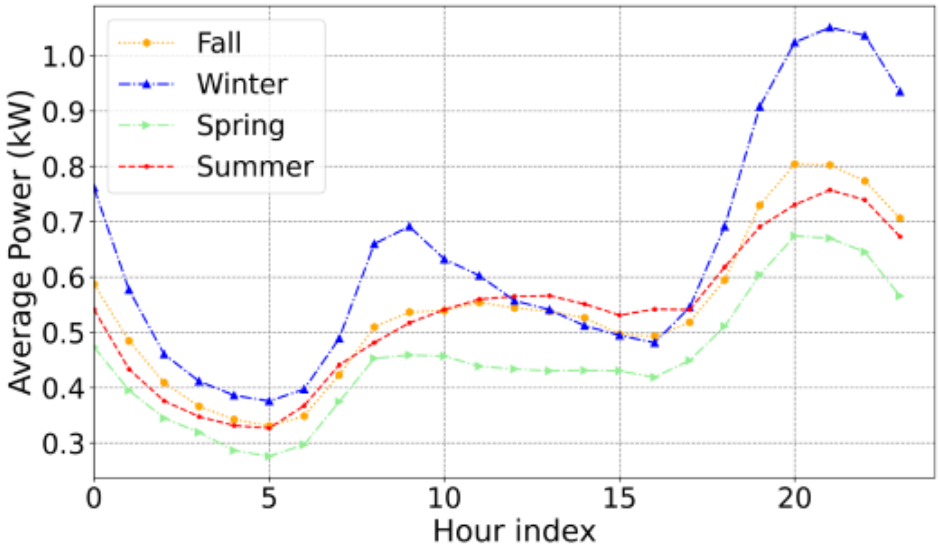


FIGURE 3-10. Average hourly power consumption crossing 100 users in different seasons.

3.5.4 Evaluation of the System's Generalization Performance

We evaluate the generalization capability of the proposed method. The term "generalization" means applying the final ANN model that has been collaboratively trained by a number of users to predict loads of new users with no historical load data.

In this case, ANN cannot be trained by the IL-based STLF method.

For this evaluation, we apply the proposed system, the AFL-based, and the FedAvg-based method to the 100 users for training and apply the trained global ANN model to another 20 new users extracted from the SGSC dataset. Table 3-3 shows the STLF comparison of the three methods for the 20 new users. Just as the observation from the result in Section 3.5.3, Table 3-3 shows that user clustering also plays an essential role in the system's generalization. When no user clustering is applied (i.e., the AFL- and FedAvg-based methods), the STLF models cannot be effectively used to generate good STLF results for the new user. When user clustering is integrated, the model training can be fast converged, and the trained global ANN model can be well used to provide STLF service to the new users. This is reflected in lower RMSE and MAPE values in Table 3-3 and also shown in Figure 3-11, which compares three methods in terms of MAPE metrics.

TABLE 3-3. Comparison of STLF under different methods on 20 new users

STLF methods	*MAPE*	*RMSE*
AFL-based method	98.62	0.245
Proposed system with K=2	89.31	0.241
Proposed system with K=3	**79.68**	**0.241**
FedAvg-based method	92.04	0.245

Figure 3-12 provides an STLF comparison of those three methods for the new users. The figure shows that the proposed system exhibits satisfactory forecast results, providing better results than the other two methods and closely following the major trend, peak values, and sudden transition of the actual load profile.

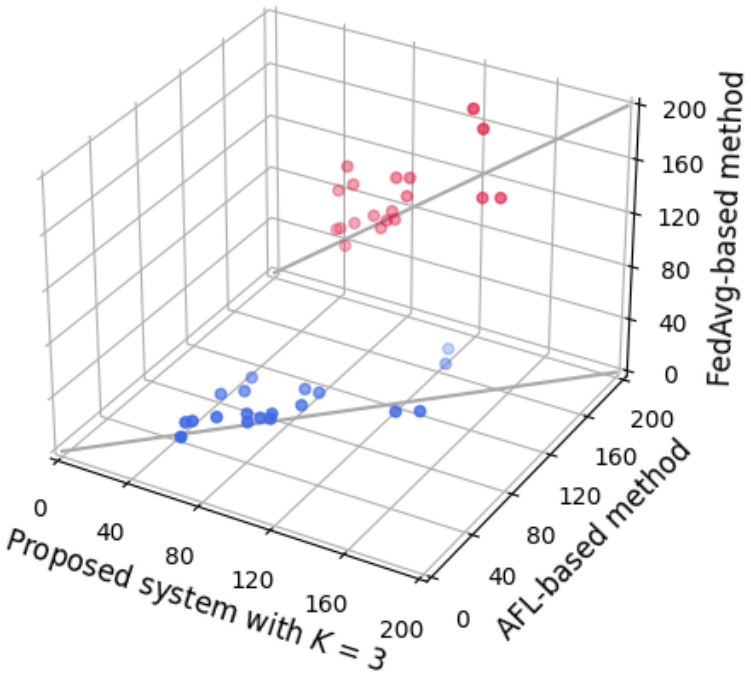


FIGURE 3-11. Comparison of different methods in MAPE on the 20 new users.

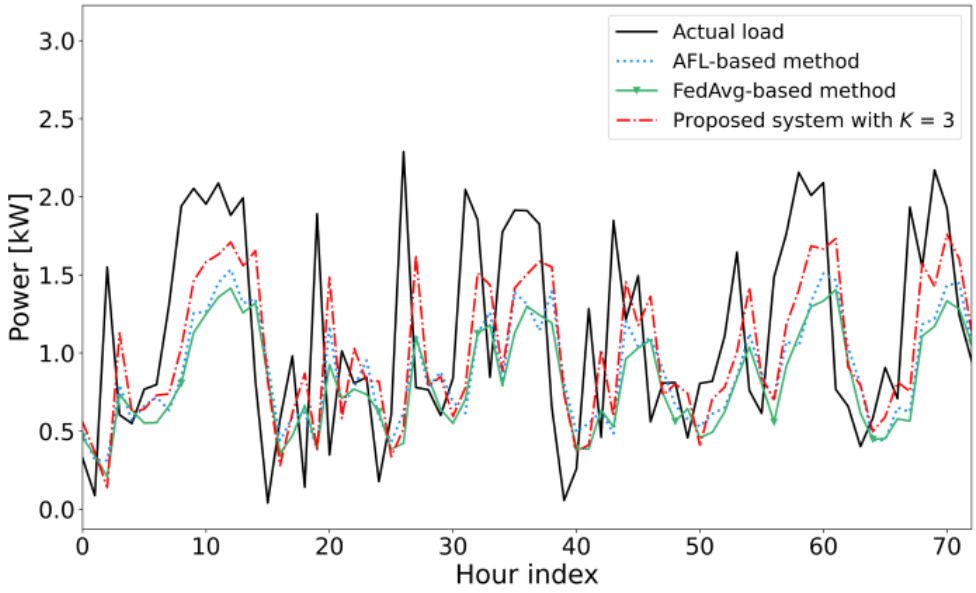


FIGURE 3-12. Demonstration of 72-hour STLF result for a new user.

3.5.5 Validation of the System's Communication Fault Robustness

As discussed in Section 3.4, one notable feature of the proposed system is that it implements asynchronously federated STLF model training. For each user cluster, the

CH does not need to wait for receiving all the local ANN parameters from all the selected users in its cluster to update the cluster ANN; instead, it updates the cluster ANN immediately once it receives the local ANN parameters from a user. This design can significantly enhance the system's robustness, subject to communication delays and faults. To validate the system's performance in this respect, we simulate a communication environment with random message loss. That is, in each communication iteration, we assign a message loss probability to each user node, representing the probability that the local ANN parameters sent by the user to the corresponding CH will be lost. Then in each communication iteration, a random number is generated in the simulation program, which is compared with the message loss probability to determine if the parameters will be lost in that iteration.

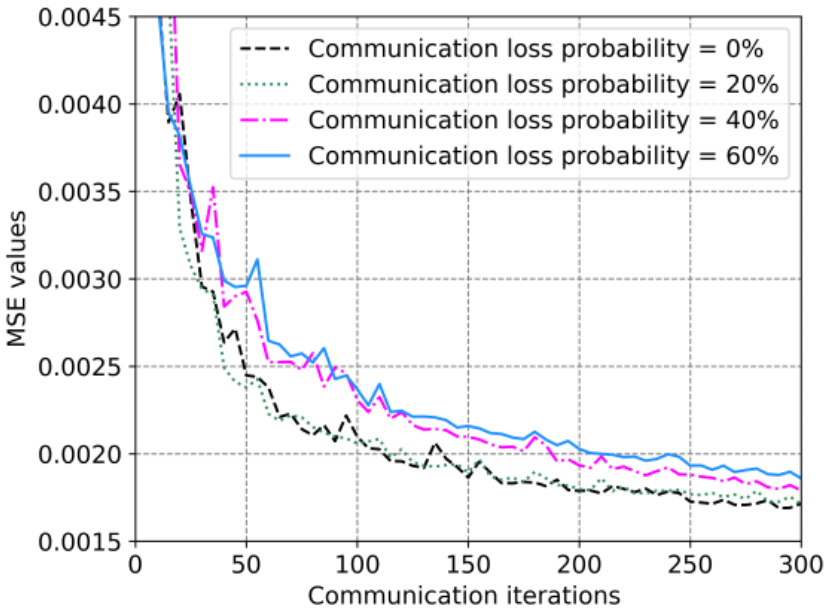


Figure 3-13. STLF performance of the proposed framework with different communication loss probabilities.

Figure 3-13 shows the convergence curves of training the global ANN model under the proposed system with different communication loss probabilities. When the communication loss probability is 20%, the model training process is not significantly affected; with the increase of the communication loss probability, the convergence rate is slightly slowed down, but the model training process can still be completed. Overall,

those experimental results validate the robustness of the proposed system and show it can adapt well to the residential environment with communication uncertainty.

3.5.6 Sensitivity Analysis for Hyper-parameters

We conduct sensitivity analysis to study the impact of four critical hyper-parameters on the system: (i) the clustering learning rate α (in Eq. (3.10)), (ii) the outlier percentile threshold s_m^p for user clustering (in Section 3.3.3), (iii) the weighting term β for asynchronously updating a cluster ANN model (in Eq. (3.17), and (iv) the global ANN model update interval R (measured as the number of communication iterations, see Section 3.4.2).

Figure 3-14 shows the performance of the distributed user clustering process under different values of α. The result is calculated as the average of 20 independent clustering trials. The average Within Cluster Sum of Squares (WCSS) [153] is used as the metric for evaluating the clustering result. The metric (denoted as $\overline{\text{WCSS}}$) is calculated as the average distance between each load profile and the centroid of the cluster to which the load profile belongs:

$$\overline{\text{WCSS}} = \frac{1}{N^{\text{total}}} \sum_{k=1}^{K} \sum_{\mathbf{P}_{m,d} \in \boldsymbol{\mu}_k} (\mathbf{P}_{m,d} - \boldsymbol{\mu}_k)^2 \tag{3.24}$$

where N^{total} is the total number of load profiles. The smaller the $\overline{\text{WCSS}}$ value is, the better the clustering performance.

From Figure 3-14, it can be seen that when the value of α is small (e.g., between 0.6 and 1.0), increasing α will generally enhance the clustering performance. When α is larger than 1.0, further increasing of the value α will degrade the performance. As presented in Section 3.3.2, the clustering iteration terminates when one of the two criteria is met: (i) the maximum iteration time is reached; or (ii) all the updated distances between the load profiles and the centroids are smaller than a pre-specified threshold. Figure 3-15 shows the impact of different values of α on the total iteration time

needed to complete the clustering process. The figure shows that by setting α in the range of [1.0, 1.4], the minimum iteration time is achieved; however, from Figure 3-14 it can be seen that the clustering performance is not quite good when α is in this range. Figure 3-14 and Figure 3-15 show that different settings of α represent the trade-off between the clustering performance and the computation cost. The results show that [0.8, 1.0] is a value range for α to achieve a good trade-off.

Figure 3-16 shows the performance of the federated user clustering process (measured in $\overline{\text{WCSS}}$) under different values of the outlier percentile threshold s_m^p. It can be seen that the value of s_m^p is nearly linear correlated to the $\overline{\text{WCSS}}$ values of the clustering process. A smaller value of s_m^p leads to a higher clustering performance. This is because setting the threshold s_m^p to be a smaller value means more load profiles will be regarded as the outliers and will be removed from the dataset. Those removed load profiles are with large distances with the cluster centroids, leading to a lower $\overline{\text{WCSS}}$ values calculated from the remaining load profiles. However, a small value of s_m^p also means reduced load data for clustering, which would make the clustering result less meaningful. In practical applications, the setting of s_m^p should be based on trials and practical considerations. Based on the experiment results in Figure 3-16, it is recommended to set s_m^p within the range of [70%, 90%] to achieve a good trade-off.

Figure 3-17 shows the convergence processes of model training in the proposed system under the different values of β. When the value of β is large (e.g., β is set to be 0.6 or 0.9), the local ANN sent from users takes a large weight to the updated cluster ANN when updating; the experiment results indicate this makes the global ANN fast converges. When β is larger than 0.5, the different values of β have little impact on the model 's convergence rate. From the result, it is recommended to set the value of β in the range of [0.5, 0.9].

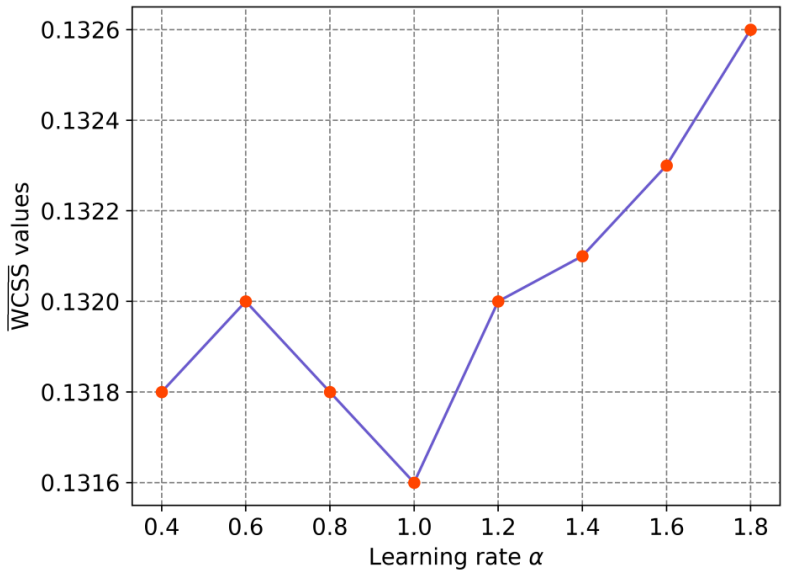


FIGURE 3-14. Performance of the federated user clustering method under different values of α.

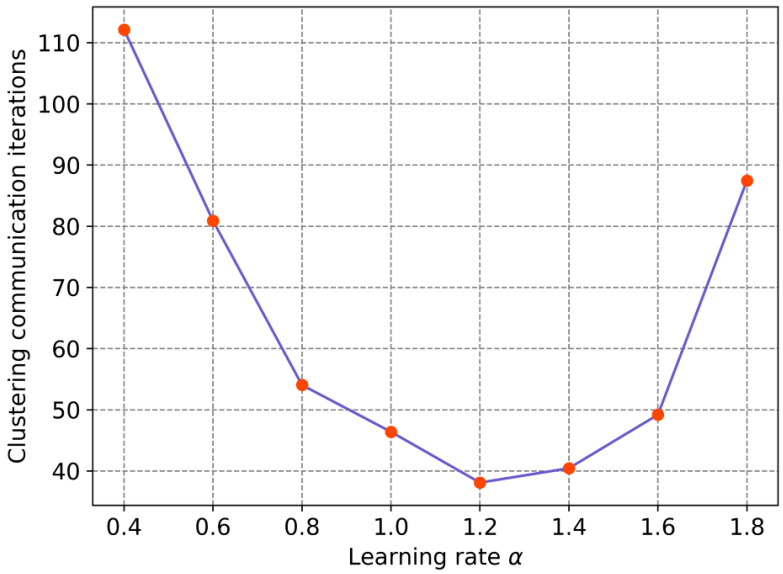


FIGURE 3-15. Number of iterations needed to complete the user clustering under different values of α.

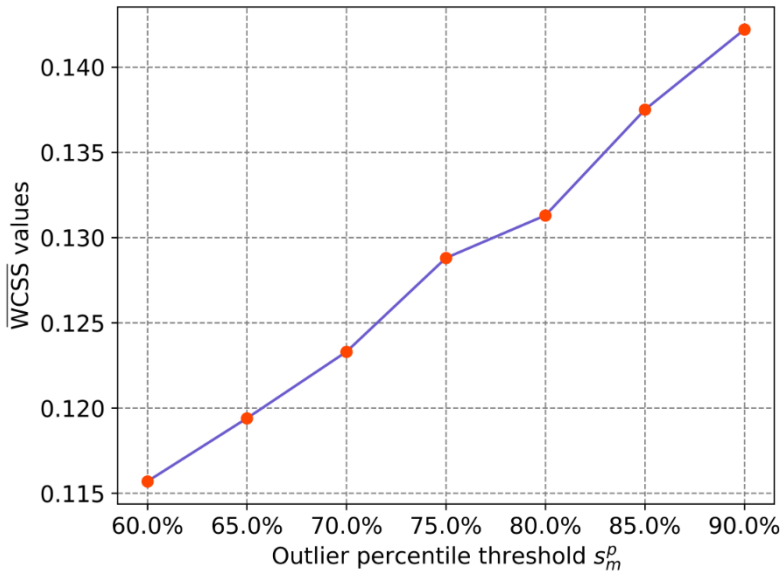


FIGURE 3-16. Performance of the federated user clustering method under the different values of outlier percentile threshold s_m^p.

Figure 3-18 shows the convergence curves of the model training process in the proposed system under different values of *R*. The figure shows that there is no significant difference in the system's performance with the different values of *R*. Updating the global model at a higher frequency (e.g., *R*=5) can slightly speed up the convergence of the model training process.

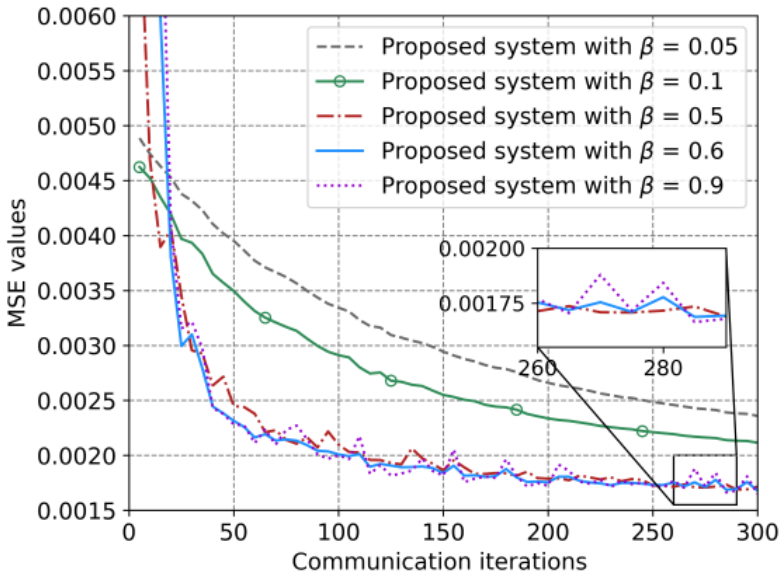


FIGURE 3-17. STLF PERFORMANCE OF THE PROPOSED FRAMEWORK WITH DIFFERENT SETTINGS OF β.

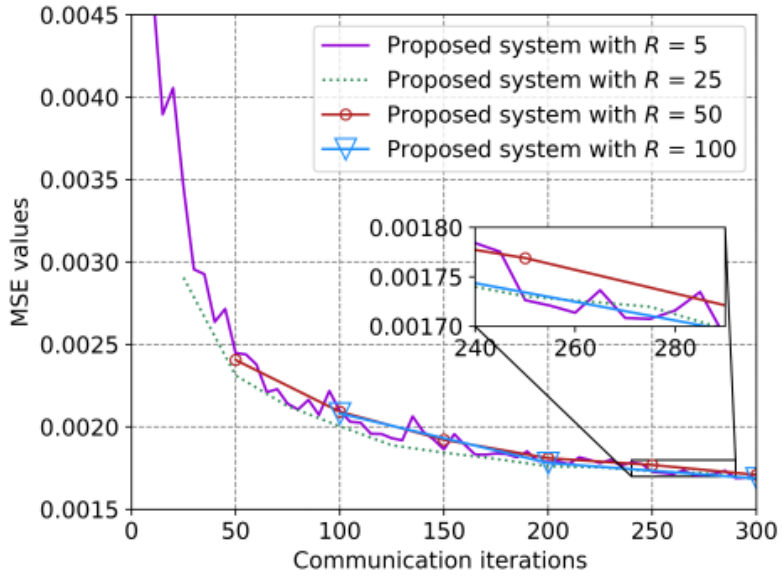


FIGURE 3-18. STLF PERFORMANCE OF THE PROPOSED FRAMEWORK WITH DIFFERENT SETTINGS OF R.

3.5.7 Evaluation of Computation Cost

Lastly, we compare the computation time of STLF under the proposed framework and the other benchmark methods. 14-day training load data of 100 users are used for the evaluation. The comparison results are shown in Table 3-4.

For the proposed method and the FedAvg-C-based method, a user clustering process is performed. It can be seen that little computation time is needed in this step for such a 100-user clustering operation (less than 2 seconds). In regard to the training time of the ANN model(s), as the table shows, the IL-based method takes the least time (only 4.77 seconds). This is because it simply uses each user's load dataset to train an ANN model for STLF and does not require communications among different users. The computation time of the ANN model training is similar to that of the other methods (including the proposed framework), i.e., less than 23 seconds. The computation time under the proposed framework is slightly larger than that under the other federated STLF methods, but the time increment (i.e., less than 2 seconds) is marginal and negligible. Since load forecasting is an ex-ante task that is typically performed on an hourly-ahead or day-ahead basis, such a computation cost is acceptable.

TABLE 3-4. Comparison of STLF computation time under different methods

STLF methods	Computation time	
	ANN training	User clustering training
AFL-based method	22.39s	Not applicable
Proposed system with K=3	22.41s	1.82s
Proposed system with K=8	22.46s	1.91s
FedAvg-C method with K=3	20.46s	1.77s
FedAvg-based method	21.07s	Not applicable
IL-based method	4.77s	Not applicable

*For the individual-based STLF method, the ANN training time is the average time used for training one local ANN model.

3.6 Chapter Summary

This chapter shows a federated learning-based privacy-preserving framework for the STLF task. This work is developed to address the limitations of conventional single-user model training, where individual users often lack sufficient data for effective training and may be unwilling or unable to share their personal energy load data with

external parties.

The proposed framework mainly integrates two key components. The first component is a federated user clustering method that groups users based on similarities in their load patterns. Instead of requiring centralized aggregation of raw consumption data, this approach only exchanges latent embeddings of user load profiles, thereby preserving privacy. By pre-clustering users, the subsequent model training becomes more efficient and converges faster compared to training without user grouping. The second component integrates a hierarchical federated learning mechanism applied after users are grouped. This mechanism enables users to collaboratively train a generic model without sharing their raw consumption data. Unlike conventional federated learning, the proposed design supports both intra-cluster and inter-cluster collaboration, allowing users with similar load behaviors to benefit from localized training while still leveraging the diversity of data across clusters. Furthermore, the framework adopts an asynchronous communication protocol. Instead of waiting for all users to complete local training before updating the global model, each local model update is immediately incorporated upon arrival at the central server. This design improves robustness against communication delays, packet losses, and user dropouts, which are common in residential network environments.

Extensive experiments are conducted in the Australian SGSC dataset to validate the effectiveness of the proposed framework. The results show that the proposed system outperforms other benchmark methods in terms of convergence speed of model training, forecasting accuracy (i.e., achieves around 37% improvement in MAPE compared with baselines), generalizability to new moved-in households, and resilience to communication faults.

CHAPTER 4

Unsupervised Load Super Resolution

This chapter proposes a new system to overcome this limitation. Supported by self-learning techniques, the system can generate load data with a desired high resolution without using high-resolution load data to train the load super-resolution model (i.e., the model can only be trained by the low-resolution load data). The system is also equipped with an unsupervised noise identification module, enabling the system to effectively identify diverse noise signals in the load sampling process and improve the reconstruction accuracy. Experiments and comparison studies are conducted on a real-world dataset to validate the effectiveness of the proposed technology, and the results demonstrate that the technology can achieve a comparable performance to supervised load super-resolution methods and has high adaptivity to load super-resolution tasks with different scaling factors.

4.1 Introduction

Load Super-Resolution (LSR) refers to a technology of reconstructing energy load (usually power consumption) from a low sampling resolution to a high resolution that can more precisely reflect the actual time series energy demand. LSR provides an economical solution for understanding the energy customers' load details using smart meters with a limited sampling capability. Because of this, LSR has been considered a fundamental supporting technology in a variety of energy applications, such as false data injection attacks and attacks in smart grids [154], abnormal load data detection [155], and load profiling [156].

4.1.1 State-of-the-Art

The simplest way of performing LSR is interpolation methods [157]. That is, using a particular interpolation strategy to generate the missing load values from their neighboring. This approach is easy to implement and computationally efficient. Nevertheless, it cannot achieve accurate load reconstruction. Recent LSR methods are based on machine learning techniques. Some methods [112, 113, 158-160] adopt the strategy of firstly transforming the time series load data into a two-dimensional (2D) form and then applying image processing techniques (typically convolutional neural networks (CNNs)) to reconstruct the transformed load data. Other methods [116, 117] apply one-dimensional (1D) CNN variants specifically designed for time series data to LSR problems. Compared with the 2D CNN-based LSR methods, the 1D CNN-based methods are lightweight, i.e., with fewer model parameters. Besides, the 1D CNN-based methods do not require transforming the load data into a 2D format, making them easier to implement and more straightforward to practice.

From the results reported in the literature [112, 113, 116, 117, 158-160], it can be seen that although CNNs and their variants have shown good performance in LSR, they are still limited in reconstructing the high-frequency details and intricate features in a load curve (e.g., sudden transitions in the load time series). To address these shortcomings, [12, 120-123] develops a generative adversarial network (GAN)-based LSR technique. In these methods, the LSR model is trained based on an adversarial training process, in which a neural network (called the “generator”) tries to create time series load data that can hardly be distinguished from the actual load data; in the meantime, another neural network (called the “discriminator”) tries to distinguish the created load data. Such competition between the generator and the discriminator can eventually enhance the generator’s ability to generate accurate, reconstructed load data. Following this principle, [124] develops an LSR method, which firstly transforms power load data into “electrical images” and then uses a GAN model to process the images and generate the LSR result. [122] proposes a GAN-based LSR method, in

which a GAN model is trained and subjected to a perceptual loss generated by a VGG-19 network [161]. The experiment results indicate that this strategy can improve load reconstruction accuracy. Despite the effectiveness of the existing methods [12, 112, 113, 116, 117, 120-124, 158-160] on LSR, they leave two open problems worth investigating.

The first open problem is associated with the data available for training the LSR model. The existing LSR techniques methods [12, 112, 113, 116, 117, 120-124, 158-160] perform load reconstruction through supervised training using low- and high-resolution load data. In this way, the model can be trained to learn the relationship between the load data and two sampling resolutions. Nevertheless, this preliminary requirement may not be satisfied in real-world applications. More commonly, an energy customer could only have load data collected by a smart meter with a low sampling resolution. In such a situation, performing LSR appears impossible due to the need for high-resolution training data. In the authors' recent work [11], we developed an LSR system based on a coordinate system that uses an autoencoder to map the load values to their timestamps. The system learns and represents the relationship between the time step coordinates of the low- and high-resolution load data. With this mechanism, the system can reconstruct low-resolution load data subjected to higher resolutions, even if the resolution of the output load data (i.e., the high resolution) is not seen in the model training data. Although the technology in [11] still requires the LSR model to be trained using load data with different resolutions, indicating the possibility of developing unsupervised LSR techniques that do not rely on high-resolution load data at the target energy customer's site.

The second open problem is associated with noise, which inevitably exists in the load data sampling process. In the existing LSR methods [11, 12, 112, 113, 116, 117, 120-124, 158-160], the noise in the low-resolution load data used to train the LSR model is assumed to follow a Gaussian distribution with known and fixed parameters. This assumption would limit the applicability of the LSR methods because, in real-world applications, it would be difficult to obtain or estimate the distribution and/or parameters of the noise. Therefore, it is necessary to develop LSR techniques that can

not only overcome the lack of high-resolution training load data but can also be well adapted to practical load sampling processes where diverse noise signals could be introduced.

4.1.2 Contributions

This study addresses the above two open problems identified from the literature. In this study, a new LSR system is proposed, which implements unsupervised LSR with adequate load reconstruction accuracy – that is, the system can be trained using only power load data sampled by a smart meter, even if the data has a low sampling resolution. The trained system can reconstruct load data at a higher resolution than the original low-resolution input during the operation stage. In addition, the system is equipped with intelligence to analyze diverse noise signals fused into the load data in an unsupervised manner, making it able to identify different noises in the load data effectively. With these abilities, the proposed system has a higher applicability in real-world scenarios compared with the methods in the literature, especially in situations where load data with high sampling resolutions at the target energy customer's site is not available or difficult to obtain to train the LSR model.

The specific contributions of this study can be summarized into the following twofold:

- To our best knowledge, this is the first research that proposes the unsupervised load super-resolution paradigm that does not require any high-resolution load data to train the LSR model. From the system implementation and numerical experiments, this study will show that by leveraging advanced self-learning techniques, implementing LSR without high-resolution training load data is not only technically feasible but can also achieve comparable and even higher load reconstruction accuracy than supervised LSR methods.
- Following the unsupervised LSR paradigm, this study develops a new LSR

system. For input load profiles with a specific low sampling resolution, the system uses the data augmentation technology to internally generate augmented variants that are with further lower resolutions and diverse noise signals; a contrastive learning process is then applied to identify augmented load profiles with similar characteristics and discriminate the ones with dissimilar characteristics. Following this, an internal learning mechanism is designed to enable the system to (i) learn the relationship between the original input low-resolution load data with the augmented load profiles with lower resolutions and noises; (ii) generalize the learned knowledge to reconstruct high-resolution load data from the input low-resolution load data.

The rest of this chapter is organized as follows: Section 4.2 provides an overview of the proposed LSR system. Sections 4.3, 4.4, and 4.5 present the technical details of the system. Section 4.6 reports the experiment result and Section 4.7 concludes the study.

4.2 Overview of LSR System

In this section, we will first formulate the LSR problem studied in this chapter, followed by the design philosophy of the proposed LSR system.

4.2.1 Problem Formulation of LSR

The objective of LSR is to reconstruct load data with a low-sampling resolution to a high-sampling resolution so that the latter can more accurately reflect the energy user's actual power consumption. Suppose a low-resolution load profile $\mathbf{P}^{lr}$ is represented as:

$$\mathbf{P}^{lr} = [P_1^{lr}, \dots, P_t^{lr}, \dots, P_{T^{lr}}^{lr}] \tag{4.1}$$

where T^{lr} is the number of load values in $\mathbf{P}^{lr}$; P_t^{lr} represents the tth load value (kW,

t=1: T^{lr}). The load values are sampled at a time interval Δt^{lr}, called the time resolution of $\mathbf{P}^{lr}$.

The high-resolution load profile (denoted as $\mathbf{P}^{hr}$) can be represented as:

$$\mathbf{P}^{hr} = [P_1^{hr}, \dots, P_p^{hr}, \dots, P_{T^{hr}}^{hr}] \tag{4.2}$$

where T^{hr} is the number of load values in $\mathbf{P}^{hr}$; P_p^{hr} represents the pth load value in $\mathbf{P}^{hr}$ (kW). The time interval between two neighboring load values P_p^{hr} and P_{p+1}^{hr} (p=1: T^{hr}-1) is Δt^{hr}, called the time resolution of $\mathbf{P}^{hr}$. There is a relationship between the two-time resolutions: $\Delta t^{lr} = \Delta t^{hr} \times \omega$, where ω is called the scaling factor, and it is a pre-specified parameter for the LSR task. Correspondingly, $T^{hr} = T^{lr} \times \omega$.

In LSR, the relationship between $\mathbf{P}^{lr}$ and $\mathbf{P}^{hr}$ can be expressed as:

$$P_t^{lr} = \frac{1}{\omega}\sum_{p=\omega(t-1)+1}^{\omega t} P_p^{hr} + \eta_t, \forall t = 1: T^{lr} \tag{4.3}$$

where η_t represents the noise associated with the tth sampled load value in $\mathbf{P}^{lr}$.

4.2.2 System Design Philosophy

As presented in Section 4.1.2, there are two primary design objectives of the proposed LSR system: (1) to facilitate LSR with the absence of high-resolution load data for training; (2) to enhance the robustness and adaptivity of the system with the presence of noises whose statistical features could be unknown.

1) Design Objective 1 – To Enable LSR without High-resolution Load Data for Training

For the first objective, such an unsupervised LSR ability could be achieved by learning the relationship between the load data with different resolutions, which inherently exists in the input, low-resolution load data $\mathbf{P}^{lr}$. Specifically, load data with an even lower resolution (denoted as $\mathbf{P}^{lr2}$) can be generated by downscaling $\mathbf{P}^{lr}$. Then, by analyzing the relationship between $\mathbf{P}^{lr}$ and $\mathbf{P}^{lr2}$, the learned knowledge can be generalized to infer $\mathbf{P}^{hr}$ from $\mathbf{P}^{lr}$. To validate the existence of such a self-similarity between the relationships of "$\mathbf{P}^{lr2} \leftrightarrow \mathbf{P}^{lr}$" and "$\mathbf{P}^{lr} \leftrightarrow \mathbf{P}^{hr}$", we examine 910 daily

load profiles with 1-minute sampling resolution across 5 U.S. residential users (source: the real-world Pecan Street dataset [111]). An average 1-day load profile is generated from the 910 daily load profiles, and this average load profile is downscaled and subjected to different scaling factors to generate load profiles with lower resolutions. For each load profile with a specific resolution, the sliding window method [162] is applied to create multiple slides, where each slide includes load values over 30 minutes. The mean gradient magnitude (MGM) [163] is calculated for each slide. A higher value of MGM indicates a larger load value variation within the slide. The probability density distribution of the MGM across all the slides is shown in Figure 4-1. Further, Table 4-1compares the Shannon entropy calculated for the average load profiles with different resolutions. A higher value of Shannon entropy indicates a higher level of randomness of the load values.

TABLE 4-1. Comparison of Shannon Entropy with different scaling factors

Scaling factor	Shannon entropy
$\omega = 1$	5.58
$\omega = 2$	5.61
$\omega = 3$	5.64
$\omega = 4$	5.65
$\omega = 5$	5.68
$\omega = 6$	5.70

Figure 4-1 shows that for load profiles with different resolutions, the probability density distribution of MGM exhibits similar shapes, e.g., the peak occurs when MGM is slightly larger than zero. Also, Table 4-1 shows that although fewer load details are contained in low-resolution load profiles, the Shannon entropy does not vary much for load profiles with different resolutions (it only slightly increases with the increase of ω). These observations indicate that the basic load variation pattern is retained in load

profiles with different resolutions. This supports the idea that one can generalize the knowledge learned from the relationship between $\mathbf{P}^{lr}$ and $\mathbf{P}^{lr2}$ to reconstruct $\mathbf{P}^{hr}$ from $\mathbf{P}^{lr}$.

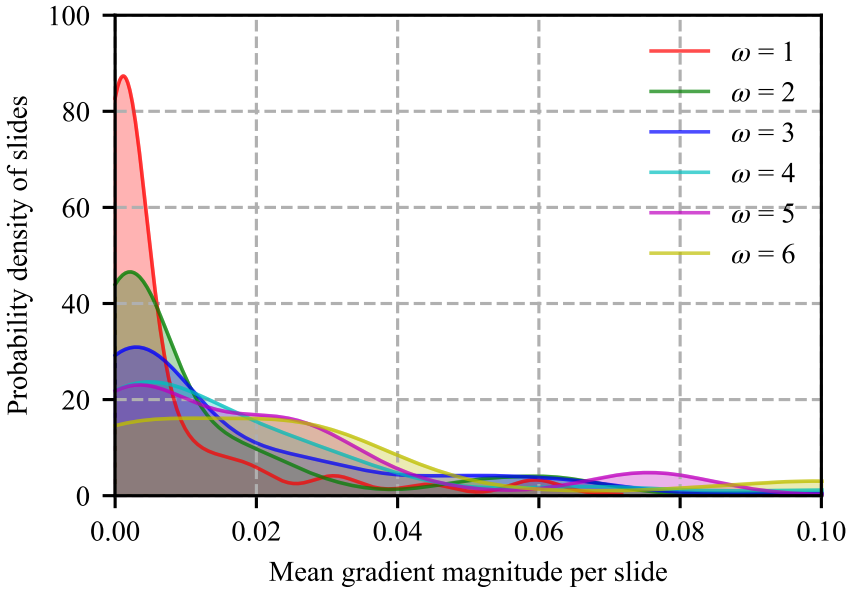


FIGURE 4-1. Probability density distribution of mean gradient magnitude per slide with different scaling factors.

Our investigation reveals that such a load resolution self-similarity between different scales can also be observed in the embeddings generated by artificial neural networks on load data. As an example, we train a tree-layer, 1D CNN model for LSR purposes using 10,000 load profiles in the Pecan Street dataset. In the model testing stage, we use this trained CNN model to perform two LSR tasks: (1) recover load profiles with a length of 360 (i.e., 360 load values in a load profile) to load profiles with a length of 1,440. This corresponds to LSR from $\mathbf{P}^{lr}$ to $\mathbf{P}^{hr}$. And (2) recover load profiles with a length of 90 to the ones with a length of 360. This corresponds to LSR from $\mathbf{P}^{lr2}$ to $\mathbf{P}^{lr}$. Figure 4-2 shows the stacked embeddings of the 1st, 2nd and 3rd layers in the CNN respectively in these two cases, where the top 3 subfigures are for the case with 360 input load data values and the bottom 3 subfigures are for the case with 90 input load values. Visually, Figure 4-2 shows a high similarity between the

embeddings generated from different resolutions of load data in each layer. The average cosine similarities between the embeddings are 0.87, 0.84, and 092 for the 1st, 2nd, and 3rd layers, respectively. Such strong similarities suggest the model captures similar essential features despite the changes of input data resolution.

The findings from the above experiments (in Table 4-1 and Figure 4-2) form the primary motivation of this research, and we will show in later sections that through properly designing the LSR system to learn the self-similarity between different load data resolution pairs, effective unsupervised LSR can be achieved.

2) Design Objective 2 – To Enable LSR with Unknown Noise Statistical Features

For the second objective, the following strategy can be applied to cope with the practical situation that usually lacks prior knowledge of the load sampling noise's statistical features. The proposed system artificially generates diverse noise signals and adds into $\mathbf{P}^{lr2}$ to generate multiple noisy load profile samples. Following this, a certain unsupervised learning mechanism can be applied to identify load profiles with similar noise signals and differentiate those with dissimilar noises. In this way, the system can be trained to learn load characteristics that could be common under different noise signals and to identify unique load characteristics caused by specific noises. This will make the system more effective in handling the input load data that could be fused with noise of arbitrary types and magnitudes.

4.3 Network Structure of LSR System

Following the design philosophy presented in Section 4.2, Figure 4-3 shows the schematic of the proposed LSR system, which includes an LSR encoder (query encoder) and an LSR generator. The former transforms the input into load data into embeddings, which are then fed into the LSR generator. The key encoder serves as an auxiliary model to support the training of the LSR encoder. The generator subsequently produces the reconstructed high-resolution load data. This section gives an overview of the model structure, and the system design will be presented in Sections 4.4 and 4.5.

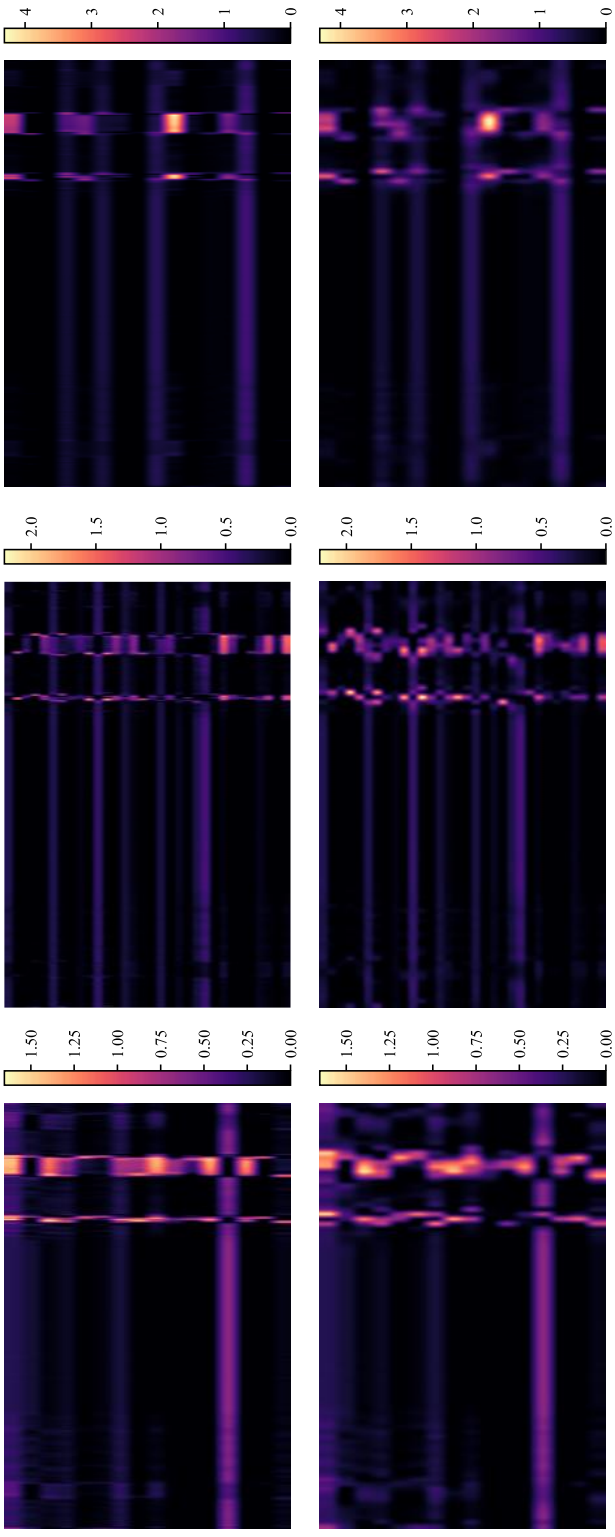


FIGURE 4-2. Stacked embeddings generated in a 3-layer CNN trained for LSR purposes. Subfigures from upper-left to upper-right: stacked embeddings in the 1st, 2nd and 3rd layers of the CNN trained by input load data with a length of 360, respectively. Subfigures from bottom-left to bottom-right: stacked embeddings in the 1st, 2nd and 3rd layers of the CNN trained by input load data with a length of 90, respectively.

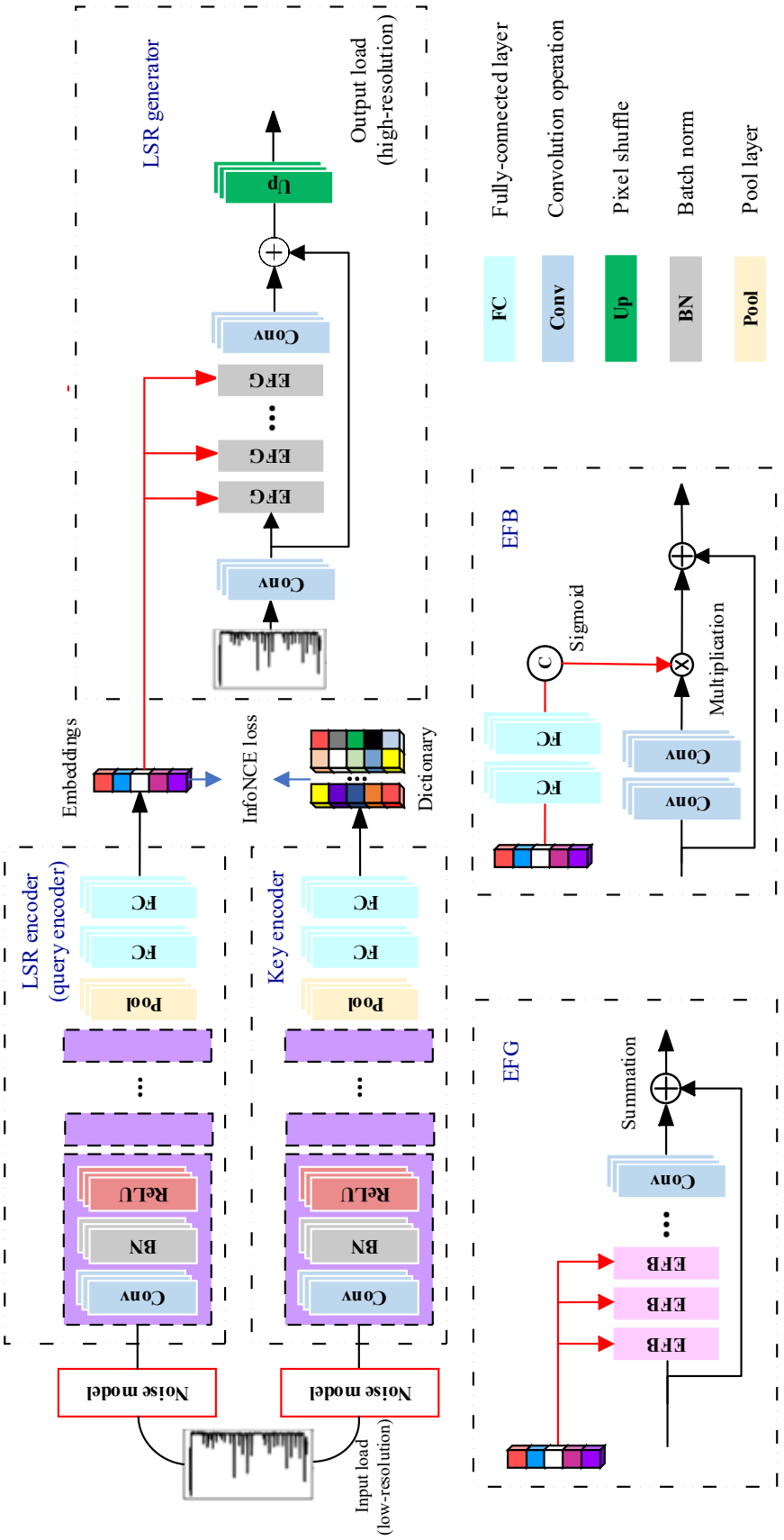


FIGURE 4-3. Architecture design of the proposed LSR system.

4.3.1 Network Structure of the Entire LSR System

The LSR encoder is a one-dimensional CNN consisting of multiple fully connected layers and convolutional layers. Each convolutional layer performs a convolution operation, batch normalization [164], and rectified linear unit (ReLU). The LSR generator comprises 3 nested modules: Embedding Fusion Groups (EFGs). Embedding Fusion Blocks (EFBs) and Embedding Fusion Modules (EFMs). Suppose a batch of training input is denoted as $\boldsymbol{\chi}^{lr}$ and is inputted into the LSR encoder and generator. The former outputs embeddings, which are denoted as $\mathbf{F}^{lr}$. The latter first processes input as:

$$\mathbf{X} = \mathrm{ReLU}\Big(\mathrm{Conv}(\boldsymbol{\chi}^{lr})\Big) \tag{4.4}$$

where $\mathrm{Conv}(\cdot)$ denotes a 1-dimensional convolutional operation; $\mathbf{X}$ is the output embeddings outputted. Next, $\mathbf{X}$ will be further processed by the EFGs in a sequential manner. For the hth EFG (denoted as $F_h^{EFG}(\cdot)$, h=1:H), it takes the output from the (h-1)th EFG (denoted as $\mathbf{X}_{h-1}$) and $\mathbf{F}^{lr}$ as inputs to generate an output $\mathbf{X}_h$:

$$\mathbf{X}_h = F_h^{EFG}(\mathbf{X}_{h-1}, \mathbf{F}^{lr}), \forall h = 1{:}\,H \tag{4.5}$$

where $\mathbf{X}_0$ denotes the output of the first convolutional layer (i.e., $\mathbf{X}$ in Eq. (4.4)). The output of the last EFG $\mathbf{X}_H$ then will be processed as:

$$\boldsymbol{\chi}^{hr} = \mathrm{Conv}\Big(F^{up}\big(\mathrm{Add}(\mathbf{X}, \mathbf{X}_H)\big)\Big) \tag{4.6}$$

where $\mathrm{Add}(\mathbf{X}, \mathbf{X}_H)$ denotes the element-wise sum operation for $\mathbf{X}$ and $\mathbf{X}_H$, which is performed as a residual connection to prevent gradient vanishing. $F^{up}(\cdot)$ denotes a shuffle operation [165]. $\boldsymbol{\chi}^{hr}$ is the output of the LSR system, representing the reconstructed load data with the required high resolution.

4.3.2 Network Structure of the LSR Generator

In the LSR generator, each EFG comprises multiple EFBs. Denoted the number of EFBs in each EFG as A; denote $F_{h,a}^{EFB}$ as the ath EFB (a=1:A) in the hth EFG. The output of one EFB is further processed by another EFB:

$$\mathbf{X}_{h,a} = F_{h,a}^{EFB}\left(\mathbf{X}_{h,a-1}, \mathbf{F}^{lr}\right), \forall h = 1{:}H,\ \forall a = 1{:}A \tag{4.7}$$

where $\mathbf{X}_{h,a}$ is the embeddings generated by $F_{h,a}^{EFB}$. $\mathbf{X}_{h,0} = \mathbf{X}_{h-1,A}$ and $\mathbf{X}_{1,0} = \mathbf{X}_0$. Each EFB comprises Z EFMs, indexed by z (z=1:Z). The network structure of EFM is based on the dynamic attention mechanism [166, 167] that modulates the importance of each embedding in the input. Each EFM performs two operations: (1) firstly, it applies the sigmoid activation function (denoted as sigmoid(·)) to $\mathbf{F}^{lr}$:

$$\mathbf{F}_{h,a,z}^{lr} = \text{sigmoid}\left(\text{FC}_{h,a,z}^{2}\left(\text{FC}_{h,a,z}^{1}(\mathbf{F}^{lr})\right)\right),$$
$$\forall h = 1{:}H, \forall a = 1{:}A,\ \forall z = 1{:}Z \tag{4.8}$$

where $\text{FC}_{h,a,z}^{1}(\cdot)$ and $\text{FC}_{h,a,z}^{2}(\cdot)$ denote the first and second full-connected layer in the sth EFM that is contained in the ath EFB and the hth EFG, respectively. $\mathbf{F}_{h,a,z}^{lr2}$ denotes the output of this EFM. (2) After performing the operation in Eq. (4.8), the EFB processes the output of the last EFB with two convolution layers and combines the processing result with $\mathbf{F}_{h,a,z}^{lr}$:

$$\mathbf{X}_{h,a,z} = \text{Relu}(\text{Conv}\left(\text{Add}\left(\mathbf{X}_{h,a,z-1}, \mathbf{F}_{h,a,z-1}^{lr} * \text{ReLu}\left(\text{Conv}(\mathbf{X}_{h,a,z-1})\right)\right)\right))$$
$$\forall h = 1{:}H,\ \forall a = 1{:}A,\ \forall z = 1{:}Z \tag{4.9}$$

where the notation "*" represents the element-wise matrix multiplication; $\mathbf{X}_{h,a,0} = \mathbf{X}_{h,a-1,Z}$. With the above operations, $\mathbf{X}_{h,a,z}$ effectively weights each embedding in $\mathbf{X}_{h,a,z-1}$, enabling the LSR system to concentrate more on critical features.

4.4 Training of LSR Encoder

The LSR encoder is first trained by a contrastive learning approach, which involves two networks with the same structure: a query encoder f^q and a key encoder f^k. After the contrastive learning process, f^q will be used as the LSR encoder; f^k only makes an auxiliary role in the contrastive learning process.

4.4.1 Load Data Augmentation and Generation of Queries and Keys

The system first scales down the input $\mathbf{P}^{lr}$ by ω to generate a load profile (denoted as $\mathbf{P}^{lr2}$) with a lower resolution than $\mathbf{P}^{lr}$:

$$\mathbf{P}^{lr2} = [P_1^{lr2}, \dots, P_s^{lr2}, \dots, P_{T^{lr2}}^{lr2}] \tag{4.10}$$

$$\omega = \frac{\Delta t^{hr}}{\Delta t^{lr}} = \frac{\Delta t^{lr}}{\Delta t^{lr2}} \tag{4.11}$$

where T^{lr2} is the number of load data items in $\mathbf{P}^{lr2}$; P_s^{lr2} denotes the sth load value (s=1: T^{lr2}). Eq. (4.11) defines that the time resolution from $\mathbf{P}^{hr}$ to $\mathbf{P}^{lr}$ is the same as that from of $\mathbf{P}^{lr}$ to $\mathbf{P}^{lr2}$. Hence, the relationship between $\mathbf{P}^{lr}$ and $\mathbf{P}^{lr2}$ is:

$$P_s^{lr2} = \frac{1}{\omega} \sum_{t=\omega(s-1)+1}^{\omega s} P_t^{lr}, \forall s = 1{:}\,T^{lr2} \tag{4.12}$$

Afterwards, the system performs data augmentation to $\mathbf{P}^{lr2}$. This is implemented by applying a noising operation to $\mathbf{P}^{lr2}$:

$$\overline{\mathbf{P}} = \mathbf{P}^{lr2} + \boldsymbol{\eta} \tag{4.13}$$

where $\boldsymbol{\eta}$ represents the noise term. By randomly generating noise samples from a specific noise model to form $\boldsymbol{\eta}$, different augmented load profile variants ($\overline{\mathbf{P}}$) can be generated. During the model training, the inputs to the query and key encoders are batches of the augmented load profiles, and each batch is represented as $\overline{\boldsymbol{\chi}}_{n,b}^{q} = [\overline{\mathbf{P}}_{n,b,1}^{q}, \overline{\mathbf{P}}_{n,b,j}^{q}, \dots, \overline{\mathbf{P}}_{n,b,J}^{q}]$ and $\overline{\boldsymbol{\chi}}_{n,b}^{k} = [\overline{\mathbf{P}}_{n,b,1}^{k}, \overline{\mathbf{P}}_{n,b,j}^{k}, \dots, \overline{\mathbf{P}}_{n,b,J}^{k}]$, where N is the number of training iterations; B is the number of batches in a training iteration; J is the number of augmented load profiles in a batch. $\overline{\mathbf{P}}_{n,b,j}^{q}$ and $\overline{\mathbf{P}}_{n,b,j}^{k}$ (n=1:N, b=1:B, j=1:J) represent the jth augmented load profiles in the bth batch that are inputted into the query and key encoders at the nth training iterations, respectively.

The query and key encoders then transform $\overline{\boldsymbol{\chi}}_{n,b}^{q}$ and $\overline{\boldsymbol{\chi}}_{n,b}^{k}$ into two sets of embeddings, called "*queries*" and "*keys*", respectively. The B sets of queries and keys generated from $\overline{\boldsymbol{\chi}}_{n,b}^{q}$ and $\overline{\boldsymbol{\chi}}_{n,b}^{k}$ (denoted as $\mathbf{F}_{n,b}^{q}, \mathbf{F}_{n,b}^{k} \in \mathbb{R}^{J \times C}$) are represented as:

$$\mathbf{F}_{n,b}^{q} = f^{q}\left(\overline{\boldsymbol{\chi}}_{n,b}^{q}\right), \forall n = 1{:}\,N, \forall b = 1{:}\,B \tag{4.14}$$

$$\mathbf{F}_{n,b}^{k} = f^{k}\left(\overline{\boldsymbol{\chi}}_{n,b}^{k}\right), \forall n = 1{:}N, \forall b = 1{:}B \tag{4.15}$$

where C is the number of features in an embedding.

4.4.2 Generation of Positive and Negative Load Profile Pairs

In the contrastive learning paradigm, the term "pair" usually refers to a pair of two embeddings, one generated from the query encoder and one generated from the key encoder. In the proposed LSR system, in a batch, a pair of embeddings are considered as positive when both are generated by applying the same noise model to the same load profile:

$$\boldsymbol{\Theta}_{n,b}^{+} = \left(\mathbf{F}_{n,b}^{q}, \mathbf{F}_{n,b}^{k}\right), \forall n = 1{:}N, \forall b = 1{:}B \tag{4.16}$$

where $\boldsymbol{\Theta}_{n,b}^{+}$ denotes the bth batch of the positive embedding pairs generated from the load profiles in the nth training iteration. The aim of setting the positive pairs in this way is to train the system to distinguish noises with different types and/or different distribution parameters.

In a batch, a pair of embeddings is considered negative when the two embeddings are generated from different load profiles or different noise models. To establish negative pairs, a matrix $\mathbf{Q} \in \mathbb{R}^{K \times C}$ is constructed, called the "*dictionary*". It is used to store the keys generated from the key encoder for future use. The hyper-parameter $K \gg B$ specifies the dictionary's size (i.e., the number of embeddings that can be stored in the dictionary). In the system, negative pairs are generated by pairing query embeddings $\mathbf{F}_{n,b}^{q}$ and each set of key embeddings stored in $\mathbf{Q}$. Based on this, a negative pair generated from $\mathbf{F}_{n,b}^{q}$ and $\mathbf{Q}$ can be represented as:

$$\boldsymbol{\Theta}_{n,b}^{-} = \left(\mathbf{F}_{n,b}^{q}, \hat{\mathbf{F}}\right), \forall n = 1{:}N, \forall b = 1{:}B, \hat{\mathbf{F}} \in \mathbf{Q} \tag{4.17}$$

When the dictionary is full (i.e., there are K sets of key embeddings stored in it), a dictionary update operation will be performed. That is, the new key embeddings generated from the current batch will be inserted into $\mathbf{Q}$, while the oldest key embeddings will be removed to ensure the dictionary will not be out of capacity. This

update process can be represented as:

$$\mathbf{Q}_{n,b+1} = \frac{\mathbf{Q}_{n,b}}{\mathbf{Q}_{n,b}^{oldest}} \cup \mathbf{F}_{n,b}^{k},\ \forall n = 1{:}N, \forall b = 1{:}B-1 \tag{4.18}$$

where $\mathbf{Q}_{n,b}^{oldest}$ is the set of oldest embeddings in $\mathbf{Q}$ in the *n*th iteration and the *b*th batch; $\mathbf{Q}_{n,b} \backslash \mathbf{Q}_{n,b}^{oldest}$ represents removing $\mathbf{Q}_{n,b}^{oldest}$ from $\mathbf{Q}_{n,b}$. This update operation dynamically refreshes the dictionary $\mathbf{Q}$ with the newly generated embeddings; in this way, it ensures $\mathbf{Q}$ up-to-date and facilitates an effective contrastive learning process.

4.4.3 Contrastive Learning-based Training

Based on the generated positive pairs and negative pairs, a contrastive learning process is performed to minimize the distance among the embeddings of the positive pairs and maximize the distance among the embeddings of the negative pairs in the feature space of embeddings. Specifically, the model parameters of the query encoder, denoted as $\boldsymbol{\theta}^{q}$, are iteratively updated as:

$$\boldsymbol{\theta}_{n+1}^{q} = \boldsymbol{\theta}_{n}^{q} - \zeta \nabla \mathcal{L}\left(\boldsymbol{\theta}_{n}^{q}\right),\ \forall n = 1{:}N-1 \tag{4.19}$$

where the subscript *n* denotes the parameters of the query encoder in the *n*th iteration; ζ is the learning rate; $\nabla \mathcal{L}(\boldsymbol{\theta}_{n}^{q})$ represents the gradient of the loss function with respect to $\boldsymbol{\theta}_{n}^{q}$. The Information Noise Contrastive Estimation (InfoNCE) [168] is used as the loss function (denoted as $\mathcal{L}_{InfoNCE}$) in the system, which quantifies the similarity among embeddings:

$$\mathcal{L}_{InfoNCE} = -\frac{1}{B}\sum_{b=1}^{B} log \frac{\exp\left(\frac{\mathbf{F}_{n,b}^{q} \cdot \mathbf{F}_{n,b}^{k}}{\gamma}\right)}{\exp\left(\frac{\mathbf{F}_{n,b}^{q} \cdot \mathbf{F}_{n,b}^{k}}{\gamma}\right) + \sum_{\hat{\mathbf{F}} \in \mathbf{Q}} \exp\left(\frac{\mathbf{F}_{n,b}^{q} \cdot \hat{\mathbf{F}}}{\gamma}\right)},$$

$$\forall n = 1{:}N \tag{4.20}$$

where γ is called temperature coefficient. $\mathbf{F}_{n,b}^{q} \cdot \mathbf{F}_{n,b}^{k}$ means performing a dot product operation for the two embeddings; $\exp(\cdot)$ is the exponential function. After updating $\boldsymbol{\theta}^{q}$, the parameters of the key encoder (denoted as $\boldsymbol{\theta}^{k}$) are updated. Since both encoders have the same network structure, $\boldsymbol{\theta}^{k}$ can be updated based on $\boldsymbol{\theta}^{q}$:

$$\boldsymbol{\theta}_{n+1}^{k} = \tau\boldsymbol{\theta}_{n}^{k} + (1-\tau)\boldsymbol{\theta}_{n}^{q},\ \forall n = 1{:}N-1 \tag{4.21}$$

where τ is a momentum coefficient.

Algorithm 4.1 summarizes the overall training process for the LSR encoder. The training process is performed for *N* times.

ALGORITHM 4.1 Procedure of LSR Encoder Training

Inputs: N, γ, τ, η, B, C, K, and training load dataset $\mathcal{D}$
Output: $\boldsymbol{\theta}^{q}$

```
Randomly initialize the dictionary Q;
For n = 1:N, Do
    Apply Eq. (4.14) to generate queries F_n^q from D;
    Apply Eq. (4.15) to generate keys F_n^k from D;
    Compute positive logits pos = F_n^q × F_n^k;
    Compute negative logits neg = F_n^q × Q_m;
    Set lg = cat(pos, neg);
    Set lables = 0_B;
    Set L = CrossEntropy(lg/γ, labels);
    Update θ_n^q following Eq. (4.19);
    Update θ_n^k following Eq. (4.21);
    Update Q following Eq. (4.18);
End For
Output θ^q
```

*"*cat*(·)" means the concatenation operation; "*CrossEntropy*(·)" represents the cross entropy operation; $\mathbf{0}_B$ represents a column vector with *B* zero elements.

4.5 Unsupervised LSR Model Training

The trained query encoder f^q (see Section 4.4) is used as the LSR encoder; it will be further trained together with the LSR generator through an internal learning process. In the following, we denote the entire LSR model as f^c; denote the LSR model's parameters as $\boldsymbol{\theta}^c = \{\boldsymbol{\theta}^q, \boldsymbol{\theta}^g\}$, where $\boldsymbol{\theta}^q$ and $\boldsymbol{\theta}^g$ represent the LSR encoder' and LSR generator's parameters, respectively.

The entire LSR model will be trained using $\mathbf{P}^{lr2}$ that is internally generated from $\mathbf{P}^{lr}$ as the input and $\mathbf{P}^{lr}$ as the output. Specifically, the LSR model is iteratively trained using batches of the input load data $\boldsymbol{\chi}_{i,\bar{b}}^{lr2} = \left[\mathbf{P}_{i,\bar{b},1}^{lr2}, \mathbf{P}_{i,\bar{b},j}^{lr2}, \ldots, \mathbf{P}_{i,\bar{b},J}^{lr2}\right]$ and the

ground truth $\boldsymbol{\chi}_{i,\bar{b}}^{lr} = \left[\mathbf{P}_{i,\bar{b},1}^{lr}, \mathbf{P}_{i,\bar{b},j}^{lr}, \ldots, \mathbf{P}_{i,\bar{b},J}^{lr}\right]$, where $\bar{B}$ and I is the number of batches and training iterations, respectively ($i = 1{:}I, \bar{b} = 1{:}\bar{B}$). The total loss $\mathcal{L}_{total}$ is defined as a combination of the Mean Squared Error (MSE, denoted as $\mathcal{L}_{MSE}$) and the InfoNCE (denoted as $\mathcal{L}_{InfoNCE}$):

$$\mathcal{L}_{total} = \mathcal{L}_{MSE} + \alpha \mathcal{L}_{InfoNCE} \tag{4.22}$$

$$\mathcal{L}_{MSE} = \left\| f^{c}\left(\boldsymbol{\chi}_{i}^{lr2}\right) - \boldsymbol{\chi}_{i}^{lr} \right\|_{2}^{2}, \ \forall i = 1{:}I \tag{4.23}$$

$$\mathcal{L}_{InfoNCE} = -\frac{1}{\bar{B}} \sum_{\bar{b}=1}^{\bar{B}} \log \frac{\exp\left(\bar{\mathbf{F}}_{i,\bar{b}}^{lr2} \cdot \frac{\tilde{\mathbf{F}}_{i,\bar{b}}^{lr2}}{\gamma}\right)}{\exp\left(\bar{\mathbf{F}}_{i,\bar{b}}^{lr2} \cdot \frac{\tilde{\mathbf{F}}_{i,\bar{b}}^{lr2}}{\gamma}\right) + \sum_{\hat{\mathbf{F}} \in \mathbf{Q}} \exp\left(\tilde{\mathbf{F}}_{i,\bar{b}}^{lr2} \cdot \frac{\hat{\mathbf{F}}}{\gamma}\right)}, \quad \forall i = 1{:}I \tag{4.24}$$

where α is a coefficient balancing the weights of the two items in the total loss; $\tilde{\mathbf{F}}_{i,\bar{b}}^{lr2}$ and $\bar{\mathbf{F}}_{i,\bar{b}}^{lr2}$ are generated by the LSR encoder, representing embeddings of the input's variants (generated by the data augmentation operation, see Eq. (4.13)), respectively. $\mathcal{L}_{MSE}$ helps the model minimize reconstruction error, ensuring that the predicted outputs are as close as possible to the actual low-resolution data. $\mathcal{L}_{InfoNCE}$ enables the model to differentiate between noise. Based on the loss function, $\boldsymbol{\theta}_{i}^{c}$ are iteratively updated as:

$$\boldsymbol{\theta}_{i+1}^{c} = \boldsymbol{\theta}_{i}^{c} - \varphi \nabla \mathcal{L}_{total}(\boldsymbol{\theta}_{i}^{c}), \forall i = 1{:}I \tag{4.25}$$

where φ represents the learning rate; $\nabla \mathcal{L}_{total}(\boldsymbol{\theta}_{i}^{c})$ represents the gradient of the loss function with respect to the model parameters $\boldsymbol{\theta}_{i}^{c}$.

The trained LSR model then can be used to reconstruct load data from a low-resolution to a desired high-resolution:

$$\hat{\mathbf{P}}^{hr} = f^{c}(\mathbf{P}^{lr}) \tag{4.26}$$

where $\hat{\mathbf{P}}^{hr}$ represents the predicted high-resolution load data.

4.6 Experiments

Experiments are conducted to evaluate the proposed unsupervised LSR system. All programs are implemented in Python and are executed on a workstation with an Intel i7-12700F processor and a 24-GB GeForce GTX 3090Ti GPU.

4.6.1 Experiment Setup

We use the Pecan Street dataset [111] to evaluate the proposed system. The dataset contains 6 to 12-month smart meter readings of 73 residential users located in Austin, New York, and California, U.S. The smart meter data is collected at different resolutions, ranging from 1 second to 15 minutes. Data preprocessing is first applied to remove the load profiles with a larger number and missing values from the dataset. After data preprocessing, we then randomly select 7,400 samples for training, 2,500 for validation, and 2,500 for testing.

We use the smart meter data at a 1-minute resolution from the dataset. The original 1-minute time resolution data is used as $\mathbf{P}^{hr}$. We downscale the 1-minute resolution data with a scaling factor of 4 (i.e.,$\omega = 4$) to generate its corresponding low-resolution data $\mathbf{P}^{lr}$. All the load data values are then standardized for model training purposes:

$$\tilde{P} = \frac{P - \mu_P}{\sigma_P} \tag{4.27}$$

where $\tilde{P}$ and P represent the standardized and actual load values, respectively. μ_P and σ_P denote the mean and the standard deviation of the load values in the training dataset, respectively. To generate the training pairs, $\mathbf{P}^{lr}$ is further downscaled to produce $\mathbf{P}^{lr2}$ (see Section 4.4). In the training stage, $\mathbf{P}^{lr2}$ is then used as input to the unsupervised LSR model, and $\mathbf{P}^{lr}$ is used as the output. In the test stage, the model directly accepts $\mathbf{P}^{lr}$ as input and generates $\widehat{\mathbf{P}}^{hr}$ that matches the original 1-minute resolution.

For the phase LSR encoder training (see Section 4.4), the maximum training iteration (N) is set to be 100. The batch size (B) is set to 256; the learning rate (ζ) is set

to be 0.001; the momentum coefficient (τ) is set to be 0.999; the temperature coefficient (γ) is set to be 0.07; and the number of embeddings in the dictionary (K) is set to be 8,192. For the phase of LSR model training (see Section 4.5), the learning rate (φ) is set to be 0.001 and Adaptive Moment Estimation (Adam) is used as the optimizer; the total number of training iterations (I) is set to be 100; the batch size is set $\bar{B}$ is set to be 256. The total numbers of EFG, EFB, and EFM (denoted as H, A, and Z, respectively) are set to be 5, 5, and 2, respectively. The kernel size of the convolutional layer is set to 3. The adaptive average pooling layer generates outputs to reduce the input's dimension to one. The fully connected layer is set to be 256 and 64 in the LSR encoder and in the EFB module, respectively. The configuration of the adaptive average pooling layer ensures a consistent input dimension for the fully connected layers, maintaining a unified architecture across all scaling factors.

4.6.2 Evaluation Metrics and Comparison Methods

The Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Frequency Component Error (FCE) are used as the evaluation metrics:

$$\mathrm{RMSE} = \sqrt{\frac{1}{W^{\mathrm{total}}}\sum_{x=1}^{W^{\mathrm{total}}}(P_x - P_x^{\mathrm{pred}})^2} \tag{4.28}$$

$$\mathrm{MAE} = \frac{1}{W^{\mathrm{total}}}\sum_{x=1}^{W^{\mathrm{total}}}\left|P_x - P_x^{\mathrm{pred}}\right| \tag{4.29}$$

$$\mathrm{FCE} = \left\| \delta\left(f^c(\mathbf{P}^{lr})\right) - \delta(\mathbf{P}^{hr}) \right\|_1 \tag{4.30}$$

where P_x and P_x^{pred} denote the xth pair of actual and reconstructed load values. $\delta(\cdot)$ denotes the Discrete Fourier Transform operation. The RMSE and MAE compare each predicted data point against its actual counterpart, and the FCE measures the frequency-domain similarity between two load profiles. The smaller the RMSE, MAE, or FCE, the higher the LSR accuracy.

The proposed system is compared with the following 4 methods:

(1) A nearest interpolation method generates values in the high-resolution load profile by linearly interpolating between the closest load values from the low-resolution

profile. It represents a simple LSR implementation and is used as a baseline in the experiments.

(2) A Super-Resolution Perception-based Incremental Learning Approach (SRP-ILA)-based method [117]. It trains a deep residual learning CNN [169] to do LSR.

(3) A Super Resolution Perception CNN (SRPCNN)-based method, which trains a 1-dimensional CNN for LSR tasks. This method is reported in [118].

(4) An LSR generator-only method. It performs LSR only using the LSR generator (in Section 4.5) without using the LSR encoder that is pre-trained following the contrastive learning method in Section 4.4.

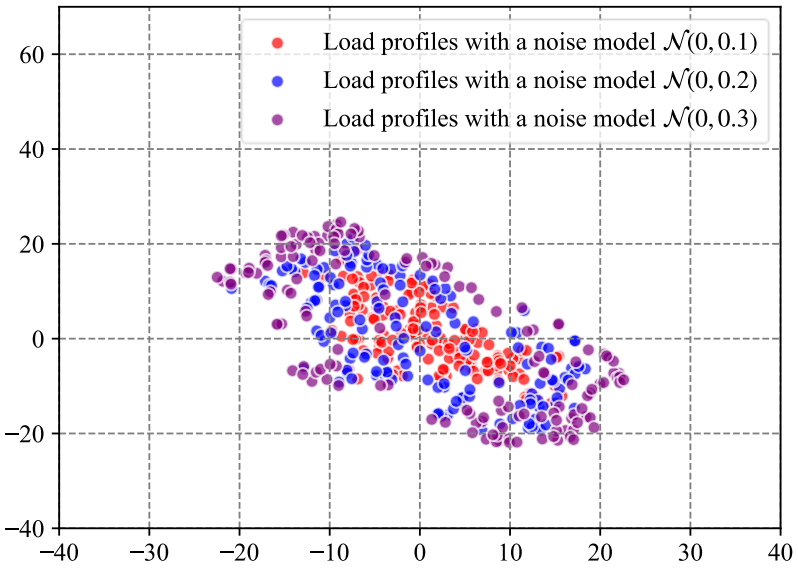
It should be noted that the SRP-ILA- and SRPCNN-based methods are supervised learning methods, meaning that they need to be trained using low-resolution load data and the load data with the required high resolution. In this way, these methods are expected to generate LSR results with adequate accuracy. In contrast, the proposed system and the LSR generator-only method perform LSR in an unsupervised manner, and it is assumed that the load data with the required high resolution is unavailable when training the models. Based on this, the comparison study aims to evaluate how the LSR result generated by the proposed method can get close to the results generated by the supervised learning-based methods.

4.6.3 Evaluation of LSR Encoder Training

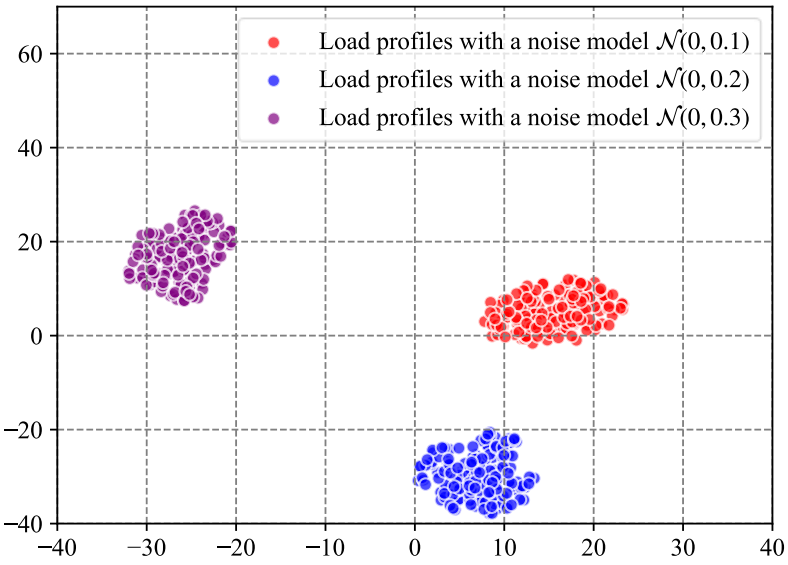
In the first part of the experiments, we evaluate the module for training the LSR encoder (see Section 4.4). Firstly, the LSR encoder is trained using the training load dataset with noise samples added. Without loss of generality, in this simulation, we set the noise model as the Gaussian distribution (denoted as $\mathcal{N}(\mu, \sigma)$, where μ and σ represent the mean and standard deviation, respectively), and in the LSR model training process, noise samples are generated with a zero mean and a standard deviation randomly varying between [0, 0.3]. In practical applications, the proposed system can utilize arbitrary noise models to generate noise samples for training the model.

The ability of the trained LSR encoder to distinguish load profiles with different noise magnitudes is then tested on 450 augmented variants generated for a specific load profile, where the augmented variants are with noises generated from three noise models: $\mathcal{N}(0, 0.1)$, $\mathcal{N}(0, 0.2)$, $\mathcal{N}(0, 0.3)$. For visualization purposes, Figure 4-4 shows the *t*-SNE results [170] without and with training the LSR encoder (for the former case, the LSR encoder is with random parameters) for the 450 augmented load profile variants. Figure 4-4 (a) shows that without training, the LSR encoder struggles to distinguish the augmented load profile variants with different noise magnitudes. On the other hand, Figure 4-4 (b) shows that the trained LSR encoder effectively clusters the embeddings generated with the same noise model, while those from different noise models are distinctly separated.

Figure 4-5 shows the convergence of the LSR encoder training process on the validation dataset, measured in terms of the accuracy of identifying positive load profile samples, i.e., the accuracy of identifying only one positive load profile sample from a dictionary consisting of 8,192 negative samples. Figure 4-5 shows that before executing the contrastive learning process (i.e., at the iteration index of zero), the LSR encoder's accuracy is quite low. With the proceeds of training iterations, the accuracy increases with small fluctuations, indicating the effectiveness of training the LSR encoder.



(a)



(b)

FIGURE 4-4. *t*-SNE visualization of the embedding of the 450 augmented load profile variants: (a) without training the LSR encoder; and (b) with training the LSR encoder.

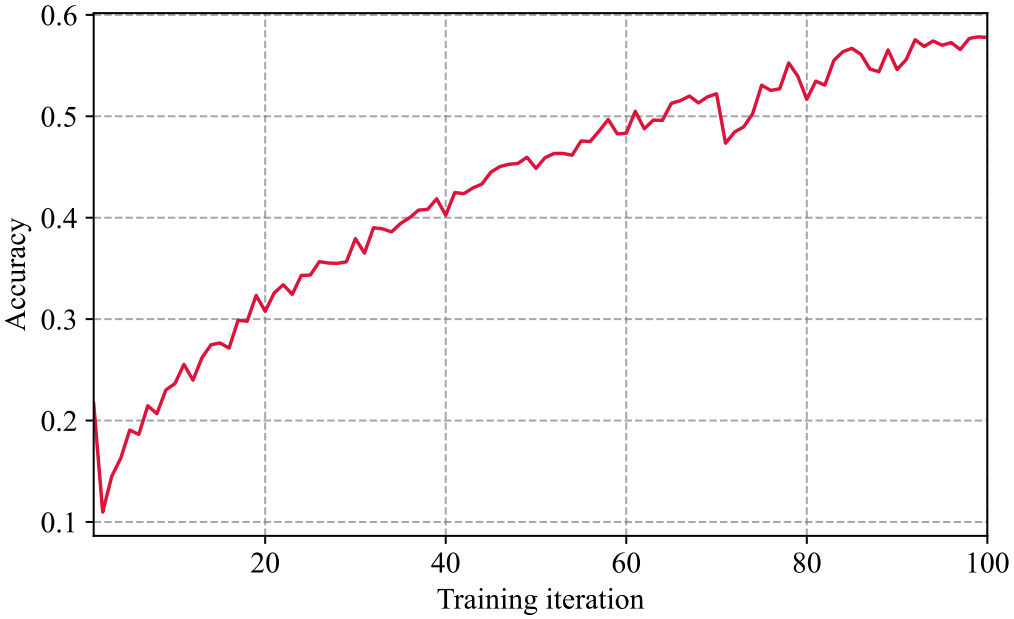


FIGURE 4-5. Convergence of the LSR encoder training process on the validation dataset.

4.6.4 Evaluation of LSR Performance

The performance of the proposed LSR system significantly depends on the internal learning mechanism (see Section 4.5). To evaluate the effectiveness of the internal learning mechanism, we compare the 4 methods in an ideal situation, i.e., the situation

without noise in the low-resolution data. The system is compared with the other 4 LSR methods introduced in Section 4.6.2, and the result is reported in Table 4-2. From the result, it can be seen that as the most straightforward method and a method not based on machine learning, the nearest interpolation method performs worst (this is reflected in the highest RMSE, MAE, and FCE values it produces among the 5 methods). In contrast, the 4 deep learning-based LSR methods (i.e., the SRP-ILA-based, SRPCNN-based, Generator-only methods, and the proposed system) perform much better. It should be emphasized that the proposed system and the generator-only method perform LSR in an unsupervised manner, while the other two perform supervised LSR. In the ideal noise-free scenario, the two supervised learning methods can generate more precise (reflected in lower RMSE, MAE and FCE values) and more stable (reflected in smaller standard deviations) LSR results than the two unsupervised methods. The better performance of supervised LSR methods over unsupervised ones is as expected, because the unsupervised LSR methods work completely without high-resolution load data. Therefore, in the experiments, we treat supervised LSR methods as baselines for evaluating the proposed method. Despite this, the results show that such a performance between supervised and unsupervised LSR methods is moderate.

TABLE 4-2. Performance comparison of load reconstruction under different methods without noise

LSR Method	Type	*RMSE*	*MAE*	*FCE*
Nearest interpolation	Non-learning	0.25	0.09	8.01
SRP-ILA-based method	Supervised learning	**0.14±0.006**	**0.06±0.001**	**4.51±0.240**
SRPCNN-based method	Supervised learning	0.15±0.005	0.06±0.002	4.79±0.242
Generator-only method	Unsupervised learning	0.17±0.008	0.08±0.002	5.17±0.213
The proposed system	Unsupervised learning	0.17±0.008	0.08±0.002	5.35±0.258

*The notation 'x±y' indicates the mean x and standard deviation y of the result obtained from 3 repeated experiments. This notation applies to the rest of the chapter.

The above interpretation of the result in Table 4-2 is further reflected in Figure 4-6,

which visualizes the LSR result of the 5 methods on a 24-hour load profile in a noise-free situation, displaying 150-time indexes. The ground truth high-resolution load data is marked as the black line in the figure. It can be seen that the unsupervised methods, including the proposed system and the LSR generator-only approach, achieve comparable LSR performance with the supervised SRP-ILA-based and SRPCNN-based methods. The proposed methods even show a better performance than the supervised methods at peak load values (around time indices 60 and 80, see the zoom-in figure in Figure 4-6). Overall, the experimental results demonstrate the effectiveness of the internal learning mechanism in LSR.

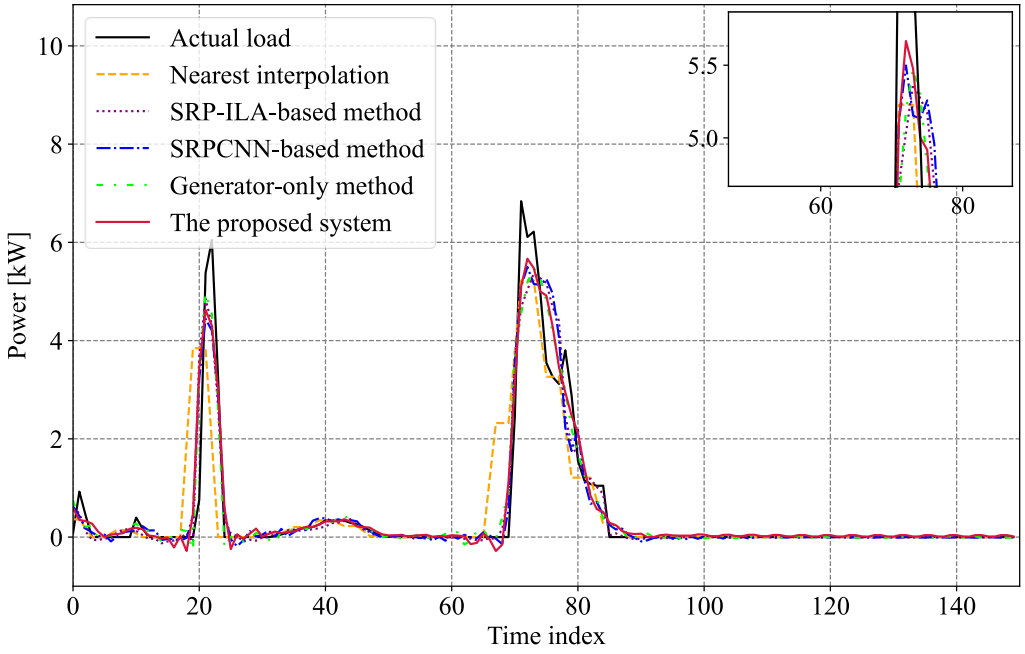


FIGURE 4-6. Reconstruction result comparison of the different methods without noise.

The proposed system is then evaluated in the presence of noise. Table 4-3 and Table 4-4 report the result of applying the trained models to perform LSR subjected to low-resolution load data with added noises that are generated from Gaussian distributions with $\sigma = 0.2$ and $\sigma = 0.3$, respectively. The results show difficulties in loading construction, with the increased amplitude of the noise, and the performance of all 5 methods degrade. Despite this, the proposed system consistently outperforms the other methods when working on the load data with different scales of noise. When considering noise, the proposed method outperforms the LSR generator-only method.

This is due to the utilization of the LSR encoder, in which the contrastive learning logic is applied to equip the system with the knowledge of understanding the characteristics of load profiles with diverse noise signals. This helps improve the system's applicability in practical situations where the noise type and magnitude could vary.

TABLE 4-3. Performance comparison of load reconstruction under different methods with noise model $\mathcal{N}(0, 0.2)$

LSR Method	Type	*RMSE*	*MAE*	*FCE*
Nearest interpolation	Non-learning	0.3664	0.2456	11.1845
SRP-ILA-based method	Supervised learning	0.31±0.007	0.23±0.007	9.04±0.349
SRPCNN-based method	Supervised learning	0.32±0.018	0.24±0.010	9.26±0.451
Generator-only method	Unsupervised learning	0.33±0.012	0.23±0.005	9.54±0.205
The proposed system	Unsupervised learning	**0.23±0.008**	**0.14±0.006**	**7.04±0.325**

TABLE 4-4. Performance comparison of load reconstruction under different methods with noise model $\mathcal{N}(0, 0.3)$

LSR Method	Type	*RMSE*	*MAE*	*FCE*
Nearest interpolation	Non-learning	0.46	0.33	13.53
SRP-ILA-based method	Supervised learning	0.43±0.023	0.33±0.007	11.80±0.517
SRPCNN-based method	Supervised learning	0.44±0.023	0.34±0.008	11.95±0.478
Generator-only method	Unsupervised learning	0.44±0.019	0.33±0.018	12.53±0.593
The proposed system	Unsupervised learning	**0.28±0.014**	**0.18±0.003**	**8.03±0.202**

Figure 4-7 shows the low-resolution load profile with noises (the dotted blue line), which consists of 360 load data items. It is used as the input of the LSR models. For reference purposes, the low-resolution load profile without noise is also shown. Figure 4-8 visually compares the results generated by the different methods to reconstruct the noised load profile in Figure 4-7 to a higher resolution load profile consisting of 1,440 data items. The ground truth 1,440-item load profile is also included in the figure as a

reference (Figure 4-8 (f)). It can be seen from Figure 4-8 that the nearest interpolation method reconstructs a rough, jagged high-resolution load profile. Also, the SRP-ILA-based and SRPCNN-based approaches do not perform better than the interpolation method with noises. Notably, the proposed system significantly outperforms the other methods in a way that the reconstructed load data closely follows the peaks and fluctuations of the ground truth load profile. In particular, comparing Figure 4-8 (d) and (e), the proposed method outperforms the generator-only method (the only difference between the two is the utilization of the LSR encoder). Consistent with Table 4-2 and Table 4-3, these results show that by integrating the contrastive learning-based LSR encoder, the proposed system can be well adapted to the noisy smart meter sampling environment and provide an affordable and effective solution to construct high-resolution load profiles with adequate accuracy.

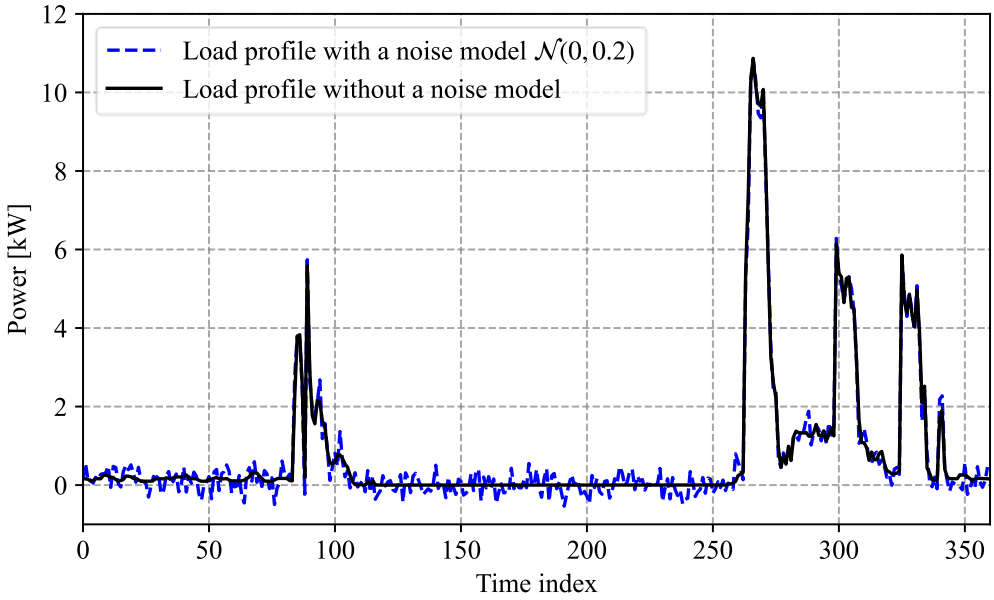


FIGURE 4-7. Comparison of low-resolution load profiles with and without noise.

Figure 4-9 compares the five methods of performing LSR tasks with different scaling factor settings (i.e., ω = 2 and ω = 3, respectively) on low-resolution load data with added noises following the noise model (i.e., σ = 0 and σ = 0.3, respectively). A larger value of the scaling factor implies a larger complexity of LSR. In the case of ω=3, the performance of all the methods degrades in terms of all the metrics MAE, RMSE, and FCE, compared with that in the case of ω=2. When there is no noise, all

the methods achieve a similar performance except for the nearest interpolation method, which performs much worse. In the scenario with noise, a large drop in the evaluation metrics is observed for the other four methods, while the proposed method performs much more robustly. This is consistent with the results in Table 4-2 to Table 4-4, showing the proposed sed system effectively handles LSR tasks across various scaling factors.

Many real-world smart meter products can only measure the load data at a relatively low temporal resolution (e.g., every 15 minutes). To evaluate the proposed method's performance in these operational environments, we perform LSRs: (1) from a 10-minute resolution to a 5-minute resolution (ω=2); and (2) from a 15-minute resolution to a 5-minute resolution (ω=3). For each of these two LSR settings, we test the system both with and without noise. The evaluation results are reported in Table 4-5. It can be seen that the proposed method delivers robust LSR results across all 4 cases, showing a comparable performance with the supervised LSR methods even with the presence of noise. This observation is generally consistent with the earlier experiments (in Table 4-2 to Table 4-4), demonstrating high applicability of the proposed method.

TABLE 4-5. Performance comparison of load reconstruction under different scaling factors and noise settings

LSR Method	From 10- to 5-minute (without noises)			From 10- to 5-minute (with noise model $\mathcal{N}(0, 0.2)$)			From 15- to 5-minute (without noises)			From 15- to 5-minute (with noise model $\mathcal{N}(0, 0.2)$)		
	RMSE	*MAE*	*FCE*	*RMSE*	*MAE*	*FCE*	*RMSE*	*MAE*	*FCE*	*RMSE*	*MAE*	*FCE*
Nearest interpolation	0.23	0.12	3.07	0.35	0.26	5.07	0.35	0.19	4.95	0.46	0.32	6.37
SRP-ILA-based method	**0.15 ±0.004**	**0.08 ±0.003**	2.08 ±0.104	0.31 ±0.018	0.23 ±0.008	4.41 ±0.182	**0.24 ±0.005**	**0.13 ±0.002**	**3.28 ±0.153**	0.39 ±0.017	0.28 ±0.010	5.34 ±0.182
SRPCNN-based method	0.15 ±0.006	0.08 ±0.003	**2.04 ±0.050**	0.31 ±0.017	0.24 ±0.010	4.48 ±0.209	0.24 ±0.007	0.15 ±0.009	3.50 ±0.111	0.40 ±0.017	0.29 ±0.006	5.52 ±0.328
Generator-only method	0.18 ±0.006	0.10 ±0.003	2.42 ±0.042	0.33 ±0.015	0.25 ±0.006	4.74 ±0.101	0.27 ±0.005	0.14 ±0.012	3.68 ±0.222	0.42 ±0.027	0.30 ±0.012	5.58 ±0.476
The proposed system	0.17 ±0.010	0.10 ±0.002	2.45 ±0.065	**0.24 ±0.006**	**0.16 ±0.006**	**3.53 ±0.197**	0.28 ±0.014	0.17 ±0.014	3.85 ±0.207	**0.34 ±0.008**	**0.23 ±0.007**	**4.71 ±0.107**

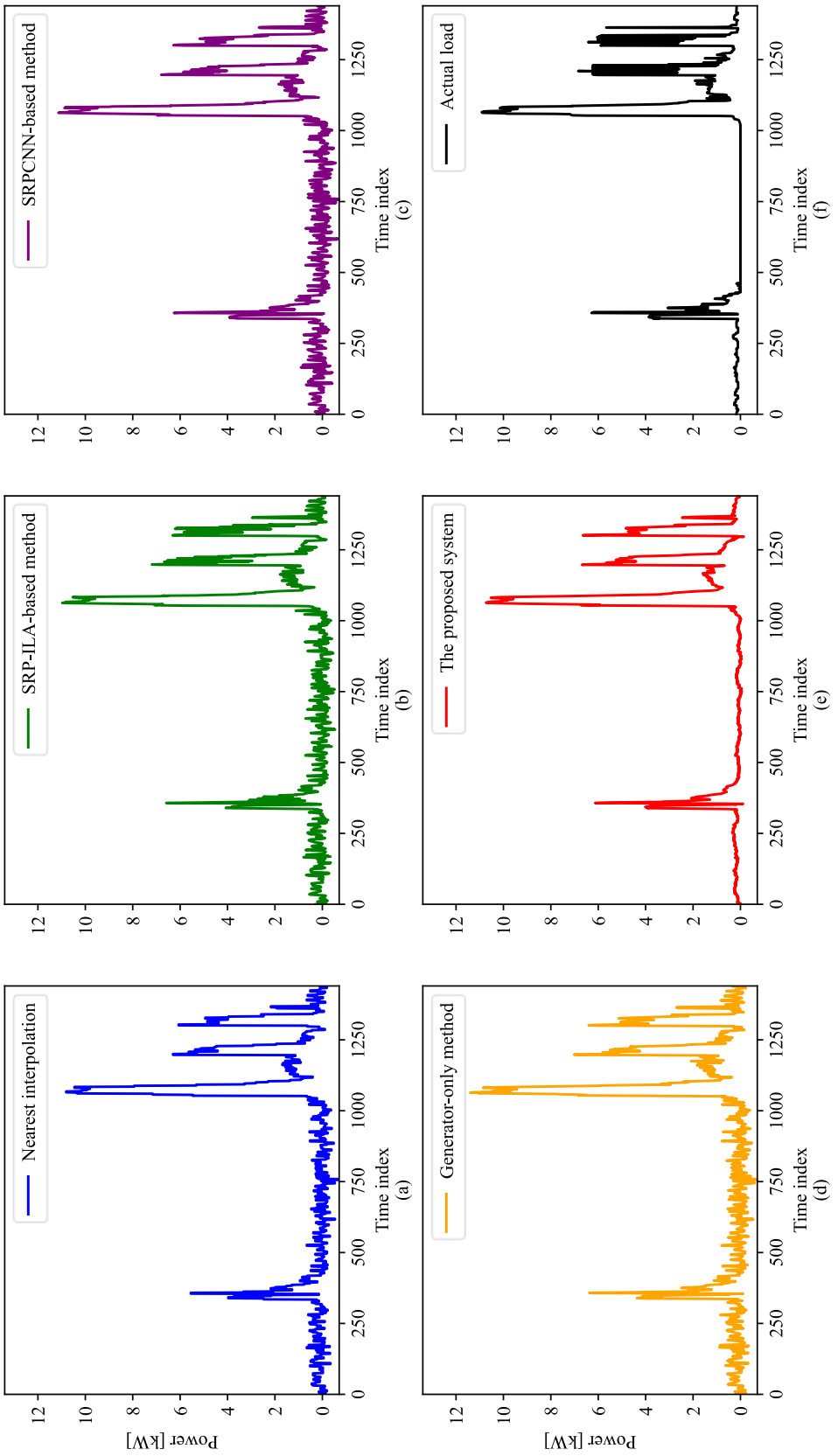


FIGURE 4-8. Comparison of load reconstruction of different methods with a noise model $\mathcal{N}(0, 0.2)$.

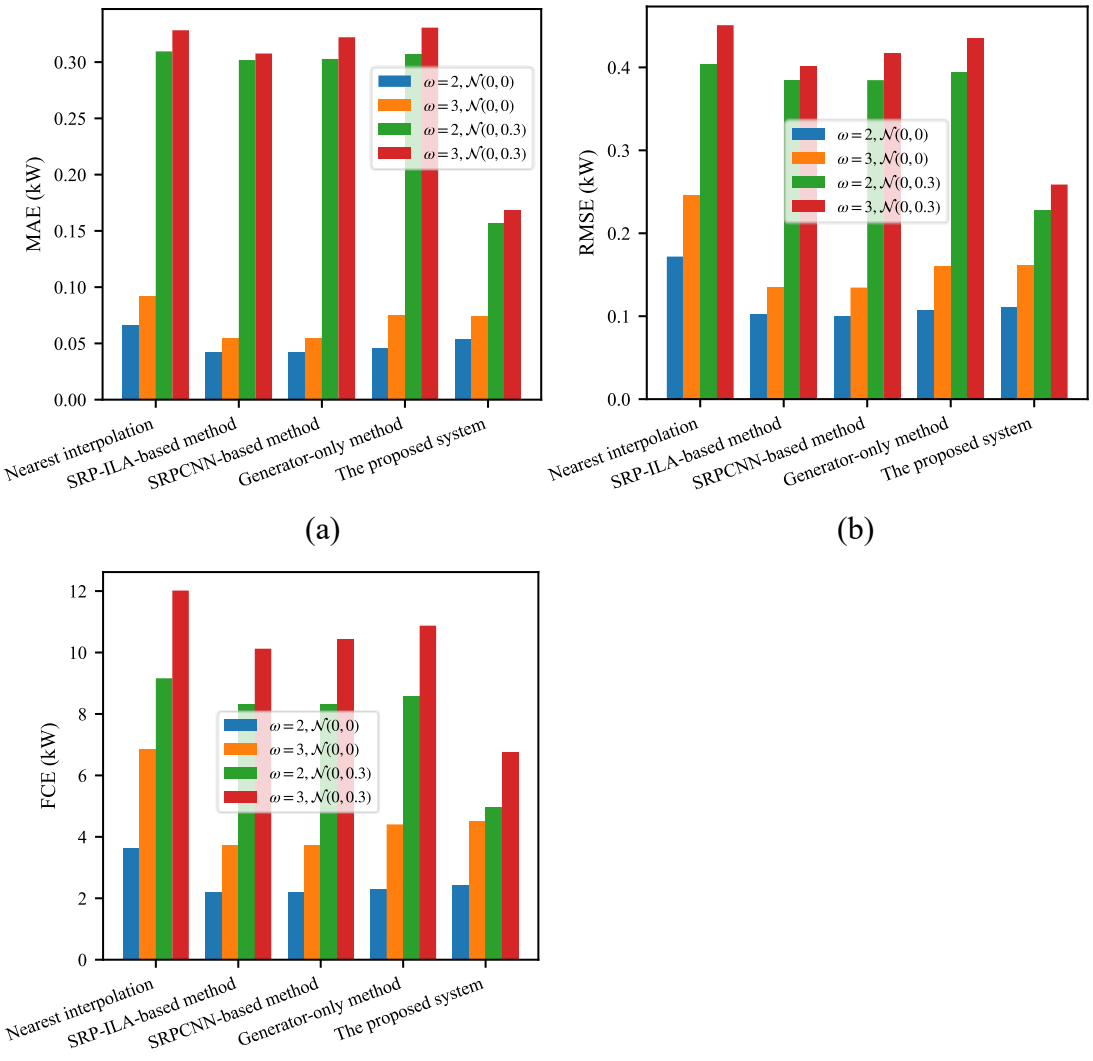


FIGURE 4-9. Performance comparison of different LSR methods under different scaling factors and noise parameters. Subfigure (a) shows average MAE values, subfigure (b) shows average RMSE values, and subfigure (c) shows average FCE values.

4.6.5 LSR Performance Evaluation under Different Parameter and Optimizer Configurations

We also test the system's performance with three different optimizer configurations: Stochastic Gradient Descent (SGD), Adam, and Root Mean Square Propagation (RMSProp). For each optimizer, the system is tested under two different learning rate settings (ζ). The system's LSR performance is measured by the MSE metric, and the results are shown in Figure 4-10. Across all 80 training iterations, all three optimizers

converge with similar MSE values. In general, the system achieves the best performance with Adam (ζ=1e−3), taking advantage of Adam's momentum-based updates [171]. In terms of convergence rate in training, the system with SGD and a learning rate of 0.001 converges the slowest, while the system with Adam and a learning rate of 0.001 converges the fastest. With the configurations of Adam (ζ=1e−3) and RMSProp (ζ=1e−4), the convergence curves show spikes of MSE during training, particularly between iterations 10 and 40 (marked by dark orange and green dotted lines in Figure 4-10); nevertheless, by reducing the learning rate by a factor of 10, smoother convergence curves can be obtained with these two optimizers. This suggests that while higher learning rates can speed up convergence, they may also introduce instability into the training process.

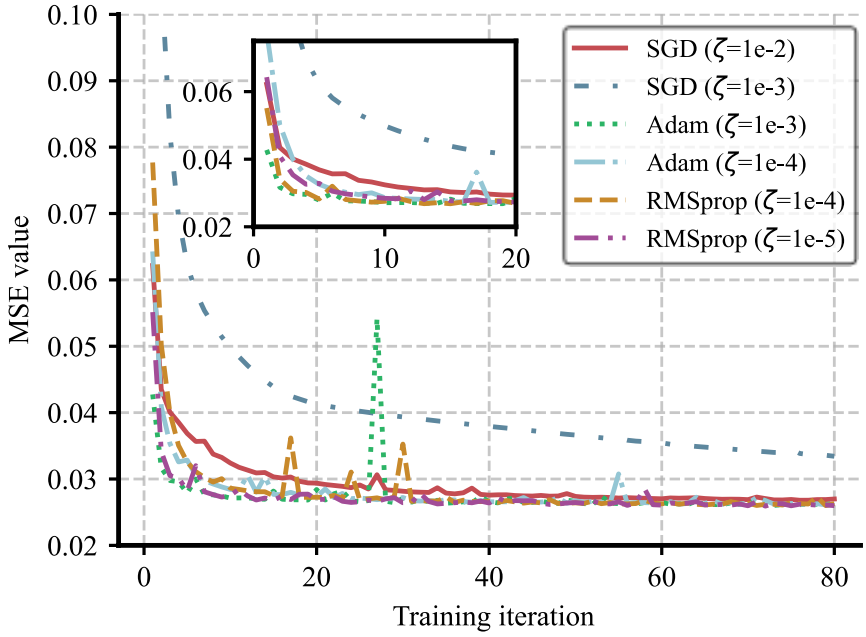


FIGURE 4-10. Convergence comparison of LSR model training under different optimizers and learning rate settings.

Further parameter sensitivity study is conducted by testing the system's performance on the test dataset with different values of EFB and EFG. When the value of one parameter (EFB or EFG) varies, the value of the other one is fixed to be 5. Figure 4-11 and Figure 4-12 show the sensitivity analysis results of these two parameters,

where the system's performance is reflected by MAE. From the results, it can be clearly seen that by setting 5 blocks for both EFB and EFG, the lowest MAE on the test dataset can be achieved. Increasing the number of blocks beyond 5 will degrade the LSR performance, and this is possibly because of overfitting caused by large values of EFB and/or EFG – an issue that could be easier to be incurred on small training datasets, such as the one used in this experiment (containing around 10,000 data pairs).

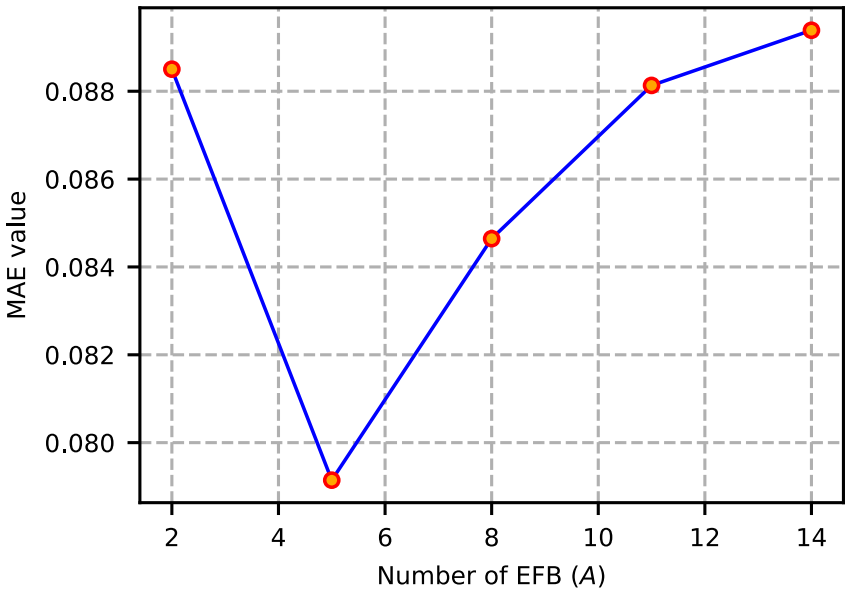


FIGURE 4-11. Performance of the proposed system under different settings of EFB.

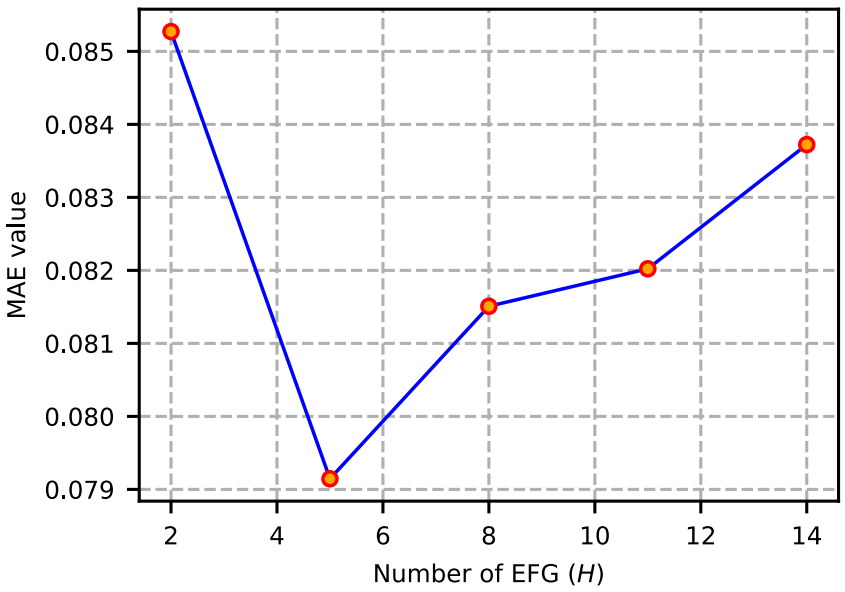


FIGURE 4-12. Performance of the proposed systems under different settings of EFG.

4.6.6 Computational Complexity Evaluation

In the last part of the experiments, we quantify the LSR methods' computational complexity in terms of: (1) number of model parameters; (2) Multiply-Accumulate Operations (MACs)[3] in model training; and (3) MACs in model testing. In this experiment, we use the Python "ptflops" tool [172] to measure the MACs, and the model testing time is measured for a batch of 64 inputs. Table 4-6 reports the evaluation results of the different methods of performing an LSR task with an input load data length of 360 with a scaling factor of 4.

For each of the supervised LSR methods (SRP-ILA-based method and SRPCNN-based method), the MACs in training and in testing are identical, because the lengths of the input load data are the same in these two stages (MACs are significantly affected by the input load data's length). The unsupervised LSR models (i.e., the generator-only and the proposed method), although they are larger than the supervised ones (i.e., with larger numbers of model parameters), they have fewer training MACs. This is because they leverage the internal learning mechanism presented in Section 4.5, which reduces the input load data length to be one- ωth of the original input data (for example, in this experiment, the input load data length in the two unsupervised LSR methods is 360/4=90). For the proposed method, MACs come from the two encoders and the generator. Each encoder uses 11.51 million MACs for training and 45.04 million MACs for testing, and the generator uses 82.3 million MACs for training and 326.46 million MACs for testing.

TABLE 4-6. Computational complexity comparison of LSR model's size

LSR Methods	Number of model parameter	MACs in training	MACs in testing
Nearest interpolation	N/A	N/A	N/A
SRP-ILA-based method	0.65M	234.57M	234.57M
SRPCNN-based method	2.30M	831.38M	831.38M
Generator-only method	1.79M	82.30M	326.46M
The proposed system	2.30M	105.32M	416.54M

*'N/A' indicates that the metric is not applicable. Metrics are not applicable to the interpolation method because it does not require model training. "M" stands for

millions.

4.7 Chapter Summary

This chapter introduces an unsupervised deep learning framework for the LSR task. Unlike conventional approaches that rely on high-resolution load data as supervisory labels, the proposed method trains the model solely using low-resolution data. It employs a self-learning strategy that leverages the inherent similarity across load profiles at different resolutions using the low-resolution dataset.

The framework is composed of two main components. The first component introduces a noise modeling mechanism designed to capture the diverse and realistic noise patterns present in residential load data. This module enhances the system's robustness against heterogeneous and noisy inputs. Specifically, it leverages a contrastive learning strategy to distinguish structurally similar from dissimilar noise instances, thereby guiding the model to learn noise-invariant representations for more reliable reconstruction. The second component adopts a self-learning strategy known as internal learning. This strategy is implemented by downscaling the available low-resolution data into even lower resolutions and then forming training pairs between the original low-resolution data and the further downscaled data. The underlying rationale, as demonstrated in this work, is that load profiles exhibit structural self-similarity across different temporal resolutions. By training the model on these lower time resolution pairs, the framework enables the reconstruction of high-resolution load profiles directly from low-resolution inputs, without requiring any external high-resolution datasets.

Experimental evaluations are conducted on the Commercial Pecan Street dataset to validate the effectiveness of the proposed method. The results show that the proposed framework achieves performance comparable to state-of-the-art supervised LSR approaches, despite relying solely on unsupervised training.

CHAPTER 5

Load Super Resolution with Arbitrary Time Resolution

This study proposes a new LSR technique. By training the LSR model only once, the system is capable of reconstructing a low-resolution load profile to be load profiles with different high resolutions that could be arbitrarily specified by the user of the system. Experiments based on a real-world dataset are conducted to validate the ability and effectiveness of the LSR system.

5.1 Introduction

Load Super-Resolution (LSR) technique refers to the technology of recovering the load data recorded by sensors with a low sampling resolution to load data with a high resolution that could reflect the actual load variation. This is achieved by expanding the single load value (e.g., a monthly load value) to multiple load values (e.g., hourly load values). LSR is meaningful for supporting energy management applications that require the load data to be with a resolution higher than the original resolution. For example, a 1-hour sampled home load profile can be recovered by LSR to be a load profile with a 15-minute resolution, so that the home energy management system can make energy trading decisions on a minute-basis a peer-to-peer energy market [173].

There have been some papers studying LSR in recent years. In [112, 121, 160], time series load data is transformed into 2-dimensional (2D) images, which are then reconstructed by Convolutional Neural Networks (CNNs). [117, 118] design CNN variants that are with customized network structures for LSR tasks. [122, 123] develop LSR systems that use Generative Adversarial Networks (GANs) to enhance the systems' ability to reconstruct sudden transitions of load profiles. [113] develops an LSR-based anomaly detection method for power load data. [116] develops an LSR method to reconstruct high-frequency components in load-photovoltaic profiles and uses the

reconstructed data for grid risk assessment. Despite the effectiveness demonstrated, these LSR methods need to retrain the LSR model every time the original load is reconstructed to a different resolution. This makes the LSR methods less computationally efficient, especially in the application scenarios where load data with multiple time resolutions need to be used at the same time.

This research study proposes a new LSR technique, which is capable of reconstructing the load data subjected to arbitrary time resolutions with only one time of model training and with sufficient preservation of the reconstruction accuracy. This cannot be achieved by the current LSR methods. The basic idea for the proposed technique to achieve this is that in the model training stage, we construct a one-dimensional coordinate system and map both the low- and high-resolution training load data to the coordinate system. That is, each load value will be assigned a unique coordinate. The LSR model will then be trained by the low- and high-resolution load data together with their coordinates, and the trained model can generate any load values at the coordinate space – or in other words, at an arbitrarily specified resolution.

This chapter is organized as follows. Section 5.2 presents the design details of the proposed LSR system. Section 5.3 reports the experiment. Section 5.4 concludes the chapter.

5.2 Design of the LSR System

5.2.1 Problem Formulation

Consider a time series load profile $\mathbf{P}^{\text{lr}}$ consisting of T^{lr} load values sampled at a regular time interval:

$$\mathbf{P}^{\text{lr}} = [P_1^{\text{lr}}, ... P_t^{\text{lr}}, ..., P_{T^{\text{lr}}}^{\text{lr}}] \tag{5.1}$$

where P_t^{lr} denotes the value of the tth sampled load value (kW, $t = 1 : T^{\text{lr}}$). A LSR task aims at generating another load profile with T^{hr} load values, denoted as

$\mathbf{P}^{\text{hr}} = [P_1^{\text{hr}}, ...P_p^{\text{hr}}, ..., P_{T^{\text{hr}}}^{\text{hr}}]$, where P_p^{hr} is the value of the pth load value (kW, $p = 1 : T^{\text{hr}}$). $\mathbf{P}^{\text{hr}}$ is with a sampling resolution ω times higher than $\mathbf{P}^{\text{lr}}$, so each load value in $\mathbf{P}^{\text{lr}}$ is the ω averaging continuous values in $\mathbf{P}^{\text{hr}}$. $\omega = T^{\text{hr}} / T^{\text{lr}}$ is the scaling factor. The relation between $\mathbf{P}^{\text{lr}}$ and $\mathbf{P}^{\text{hr}}$ can be expressed as:

$$P_t^{\text{lr}} = \frac{1}{\omega} \sum_{p=\omega(t-1)+1}^{\omega t} P_p^{\text{hr}} + \eta_t, \; t = 1 : T^{\text{lr}}, p = 1 : T^{\text{hr}} \tag{5.2}$$

where η_t denotes the noise generated in the data collection process for the tth load in $\mathbf{P}^{\text{lr}}$.

5.2.2 Overview of the Proposed LSR System

The proposed system takes the following inputs: (i) a load profile with a low resolution ($\mathbf{P}^{\text{lr}}$); and (ii) a scaling factor ω. It outputs the reconstructed high-resolution load profile $\mathbf{P}^{\text{hr}}$. This can be represented as:

$$\mathbf{P}^{\text{hr}} = f_{\boldsymbol{\theta}}(\mathbf{P}^{\text{lr}}, \omega) \tag{5.3}$$

where $f_{\boldsymbol{\theta}}(\cdot)$ represents the LSR model with parameters $\boldsymbol{\theta}$. In existing LSR methods, the LSR model needs to be trained separately for different values of ω. In contrast, the proposed system is designed for arbitrary load super-resolutions with only one model training. The fundamental mechanism for the system to achieve this is to map time series load data to a 1-dimensional (1D) coordinate system, in which a coordinate represents a time point. With this representation, reconstructing load values subjected to different super-resolutions is equivalent to generating data values at the coordinates of different intervals. Since coordinates can be established at arbitrary scales, the system can achieve load reconstruction with arbitrary super-resolutions once it is properly trained.

The schematic of the proposed LSR system is shown in Figure 5-1. The system is backboned by two encoders and one decoder. One encoder (called the "EDSR" encoder) uses an Enhanced Deep Super-Resolution (EDSR) network [174] to generate load

embeddings $\mathbf{F}$ from the input load profile $\mathbf{P}^{\text{lr}}$, and the other encoder (called the "coordinate encoder") generates embeddings $\mathbf{S}$ from the coordinates of $\mathbf{P}^{\text{lr}}$ and $\mathbf{P}^{\text{hr}}$. $\mathbf{F}$ is then interpolated to generate embeddings $\bar{\mathbf{F}}$ that has the same size with $\mathbf{S}$. $\bar{\mathbf{F}}$ and $\mathbf{S}$ are fed into the decoder ψ^{dec} to generate the load data values in $\mathbf{P}^{\text{hr}}$ as the output. As introduced before, in the proposed LSR system, a load data value is regarded as the data at a specific coordinate. Hence, the generation of the load data value at a coordinate x can be generally represented as:

$$P_x = \psi^{dec}(\bar{\mathbf{F}}_x, \mathbf{S}_x) \tag{5.4}$$

where $\bar{\mathbf{F}}_x$ is the embedding section at the coordinate x in $\bar{\mathbf{F}}$; $\mathbf{S}_x$ is the embedding at the coordinate x in $\mathbf{S}$.

The system is trained by a set of training samples. Each training sample contains a low-resolution load profile with a length of T^{lr} and a high-resolution load profile with a scaling factor ω. The value T^{lr} is the same for all the samples while different samples could have different values ω (i.e., the resolutions of the high-resolution load profiles in different training samples could be different). After the system is trained, $\mathbf{P}^{\text{hr}}$ is generated in multiple rounds, and in each round, the system generates T^{lr} data values.

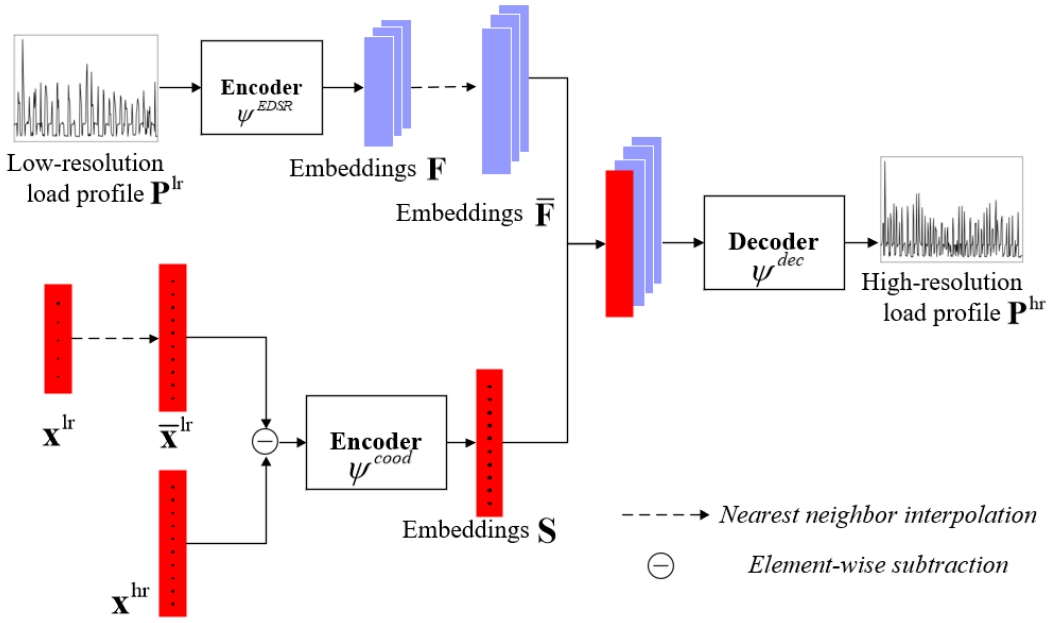


FIGURE 5-1. Schematic of the proposed LSR system.

5.2.3 Generation of Coordinate System

First, the system generates the 1D coordinate system for $\mathbf{P}^{\text{lr}}$ and $\mathbf{P}^{\text{hr}}$, respectively. Denote the sets of the coordinates of the load data values in $\mathbf{P}^{\text{lr}}$ and $\mathbf{P}^{\text{hr}}$ as $\mathbf{x}^{\text{lr}} = [x_1^{\text{lr}}, ..., x_{T^{\text{lr}}}^{\text{lr}}]$ and $\mathbf{x}^{\text{hr}} = [x_1^{\text{hr}}, ..., x_{T^{\text{hr}}}^{\text{hr}}]$, respectively, in which x_t^{lr} (t=1: T^{lr}) and x_p^{hr} (p=1: T^{hr}) represent the coordinates of the tth and pth data values in $\mathbf{P}^{\text{lr}}$ and $\mathbf{P}^{\text{hr}}$, respectively. The coordinates are within the range of [-1, 1] and are expressed as:

$$x_t^{\text{lr}} = \frac{-1 + 2t - T^{\text{lr}}}{T^{\text{lr}}}, \quad t = 1 : T^{\text{lr}} \tag{5.5}$$

$$x_p^{\text{hr}} = \frac{-1 + 2p - T^{\text{hr}}}{T^{\text{hr}}}, \quad p = 1 : T^{\text{hr}} \tag{5.6}$$

For example, consider T^{lr}=2 and $\omega = 2$, then following Eqs. (5.5) and (5.6), it can be determined $\mathbf{x}^{\text{lr}} = [\text{-}0.5, 0.5]$ and $\mathbf{x}^{\text{hr}} = [\text{-}0.75, \text{-}0.25, 0.25, 0.75]$.

5.2.4 Design of Encoders

Based on the generated coordinate, $\mathbf{P}^{\text{lr}}$ is fed into the EDSR encoder $\psi^{EDSR}(\cdot)$ to generate embeddings $\mathbf{F} \in \mathbb{R}^{B \times C \times T^{\text{lr}}}$, which consists of T^{lr} embedding sections:

$$\mathbf{F}_{x_t^{\text{lr}}} = \psi^{EDSR}(P_{x_t^{\text{lr}}}^{\text{lr}}), \; t = 1, 2, ..., T^{\text{lr}} \tag{5.7}$$

where C is the dimension of embeddings; B is the batch size; $\mathbf{F}_{x_t^{\text{lr}}}$ denotes the x_t^{lr}th embedding section in $\mathbf{F}$, which consists of $B \times C$ features for the x_t^{lr}th coordinate. Since EDSR is originally designed for 2-dimensional (2D) data (e.g., images), we propose to use 1D convolutional layers to replace the 2D layers in the EDSR network, so that it can process 1D data, i.e., the power load data in this study. The structure of

the 1D EDSR is illustrated in Figure 5-2.

A nearest neighbor interpolation operation is applied on $\mathbf{x}^{\mathrm{lr}}$ to generate a new coordinate $\bar{\mathbf{x}}^{\mathrm{lr}} = [\bar{x}_1^{\mathrm{lr}}, ..., \bar{x}_{T^{\mathrm{hr}}}^{\mathrm{lr}}]$. With the interpolation operation, $\bar{\mathbf{x}}^{\mathrm{lr}}$ consists of T^{hr} coordinates, and each coordinate $\bar{x}_p^{\mathrm{lr}}$ is determined as:

$$\bar{x}_p^{\mathrm{lr}} = x_t^{\mathrm{lr}}, \quad t = 1:T^{\mathrm{lr}}, \quad p = 1:T^{\mathrm{hr}} \tag{5.8}$$

where $t = \arg\min_{t=1}^{T^{\mathrm{lr}}} \left| x_t^{\mathrm{lr}} - x_p^{\mathrm{hr}} \right|$. Also, for $\mathbf{F}_c$ (c=1:C), a nearest neighbor interpolation operation is applied to generate a new embedding $\bar{\mathbf{F}}_c$ consisting of T^{hr} coordinates. Therefore, C new embeddings will be generated. Each embedding value $\bar{F}_{c,x_p^{\mathrm{hr}}}$ in $\bar{\mathbf{F}}_c$ is:

$$\bar{F}_{c,x_p^{\mathrm{hr}}} = F_{c,x_t^{\mathrm{lr}}}, \ p = 1:T^{\mathrm{hr}}, \ t = 1:T^{\mathrm{lr}}, \ c = 1:C \tag{5.9}$$

where $t = \arg\min_{t=1}^{T^{\mathrm{lr}}} \left| x_t^{\mathrm{lr}} - x_p^{\mathrm{hr}} \right|$, $c = 1:C$. The set of the C new embeddings is denoted as $\bar{\mathbf{F}}$. The embeddings of the coordinates $\mathbf{S} \in \mathbb{R}^{B \times 1 \times T^{\mathrm{hr}}}$ are then generated by the coordinate encoder $\psi^{cood}(\cdot)$, which is a Fully Connected-layer Network (FCN). It takes the element-wise subtraction of $\mathbf{x}^{\mathrm{hr}}$ and $\bar{\mathbf{x}}^{\mathrm{lr}}$ as the input:

$$\mathbf{S} = \psi^{cood}(\mathbf{x}^{\mathrm{hr}} - \bar{\mathbf{x}}^{\mathrm{lr}}) \tag{5.10}$$

5.2.5 Design of Decoder with a Local Regression Mechanism

The decoder takes $\mathbf{S}$ and $\bar{\mathbf{F}}$ as inputs and maps them to the output load profile. For the decoder design, we propose a Local Regression Mechanism (LRM), which works with an FCN to reconstruct the load by modelling the correlation among load values in neighboring time points. With LRM, the output load value at the x_p^{hr} coordinate (denoted as $P_{x_p^{\mathrm{hr}}}^{\mathrm{hr}}$) is generated as:

$$P^{\mathrm{hr}}_{x_p^{\mathrm{hr}}} = \left|\frac{x_p^{\mathrm{hr}} - \xi(x_p^{\mathrm{hr}})}{\tau(x_p^{\mathrm{hr}}) - \xi(x_p^{\mathrm{hr}})}\right| \cdot \psi^{dec}(cnt(\mathbf{S}_{\xi(x_p^{\mathrm{hr}})}, \bar{\mathbf{F}}_{\xi(x_p^{\mathrm{hr}})})) + \left|\frac{\tau(x_p^{\mathrm{hr}}) - x_p^{\mathrm{hr}}}{\tau(x_p^{\mathrm{hr}}) - \xi(x_p^{\mathrm{hr}})}\right| \cdot \psi^{dec}(cnt(\mathbf{S}_{\tau(x_p^{\mathrm{hr}})}, \bar{\mathbf{F}}_{\tau(x_p^{\mathrm{hr}})})),\ p = 1, 2, \ldots, T^{\mathrm{hr}} \tag{5.11}$$

where $\xi(x_p^{\mathrm{hr}})$ and $\tau(x_p^{\mathrm{hr}})$ return the x_p^{hr}th nearest left and right coordinates, respectively; $\mathbf{S}_{\xi(x_p^{\mathrm{hr}})}$ and $\mathbf{S}_{\tau(x_p^{\mathrm{hr}})}$ represent the values at the coordinates $\xi(x_p^{\mathrm{hr}})$ and $\tau(x_p^{\mathrm{hr}})$ in $\mathbf{S}$, respectively. $\bar{\mathbf{F}}_{\xi(x_p^{\mathrm{hr}})}$ and $\bar{\mathbf{F}}_{\tau(x_p^{\mathrm{hr}})}$ represent the values at the coordinates $\xi(x_p^{\mathrm{hr}})$ and $\tau(x_p^{\mathrm{hr}})$ in $\bar{\mathbf{F}}$, respectively. The function $cnt(\cdot)$ performs a concatenation operation to the inputs. $\psi^{dec}(\cdot)$ represents the decoder, which is an FCN. Model (5.11) generates each load value in $\mathbf{P}^{\mathrm{hr}}$ from its neighboring load values following a weighted average strategy. It adds weights to the neighboring load values based on the distances between them and $P^{\mathrm{hr}}_{x_p^{\mathrm{hr}}}$. The value of $P^{\mathrm{hr}}_{x_p^{\mathrm{hr}}}$ is then determined as the sum of the weighted neighboring load values.

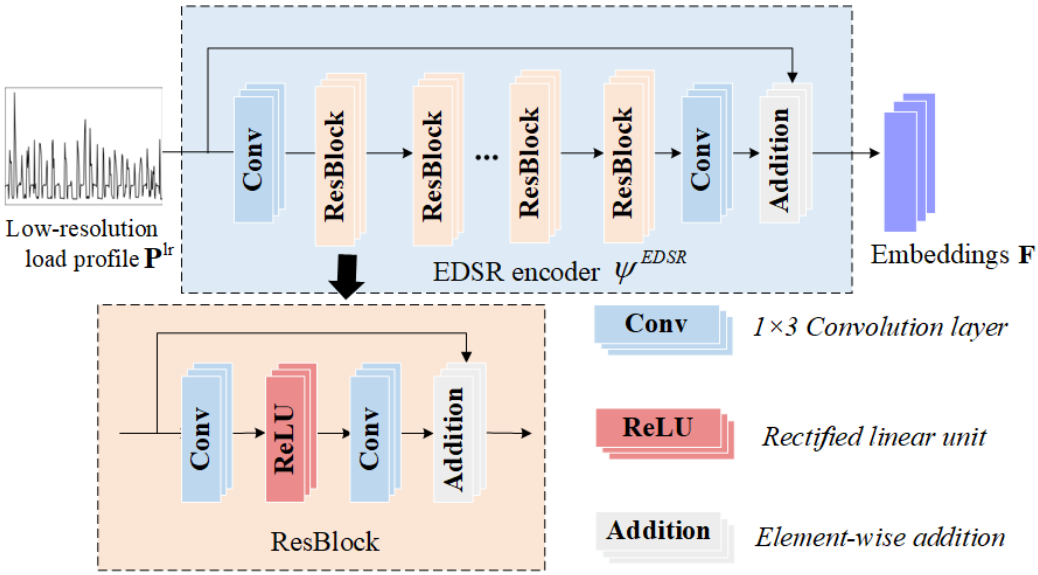


FIGURE 5-2. Network structure of 1D EDSR encoder.

5.3 Experiments

5.3.1 Experiment Setup

The LSR system is implemented by Python and is validated on the Pecan Street dataset [111], which contains smart meter readings of 75 households in the USA. The readings cover 6 to 12 months and are taken with a 1-minute sampling interval. We organize the smart meter data into daily load profiles, resulting in 6,500 load profiles. The load profiles of 60, 5, and 10 households are used for training, validation, and testing, respectively. The load profiles are down-sampled to be with a 30-minute resolution, which are used as the low-resolution input data of the LSR system. In this way, T^{lr} is set to be 48. Gaussian noises with a zero mean and a variance of 0.01 are added into $\mathbf{P}^{\text{lr}}$. The batch size is set to 64. During the training, in each batch, 64 scaling factors (i.e., ω_b, b=1:64) are randomly generated within [1, 4], and 48 load values are randomly selected from the $48\times\omega_b$ high-resolution load values to form the fixed-length output.

The Mean Absolute Error (MAE) is used as the loss function to train the proposed method. The encoder is set to contain 32 residual blocks (i.e., the "ResBlock" in Figure 5-2). Each convolution layer contains 256 channels. $\psi^{cood}(\cdot)$ and $\psi^{dec}(\cdot)$ are composed of 3 fully connected layers, and each layer has 256 neurons. The learning rate is set to be 0.0001 and the model is trained for 100 epochs.

5.3.2 Results and Discussion

We compare the proposed system with 7 benchmark methods that can be used for LSR: (i) the nearest interpolation method, (ii) the cubic interpolation method, (iii) the Back Propagation-Artificial Neural Network (BP-ANN) method, (iv) the Super Resolution Perception CNN (SRPCNN) method [118], and (v) the EDSR-2D method [174], which is commonly used in 2D image reconstruction. To validate the

effectiveness of the coordinate encoding scheme and the LRM, we further compare the proposed method with: (vi) the EDSR-1D method, and (vii) the proposed method without using LRM (denoted as the "None-LRM" method). The Root Mean Square Error (RMSE), MAE, and Critical Point Error (CPE) [122] metrics are used to evaluate the load reconstruction performance.

Table 5-1 reports the load reconstruction performance generated by the different methods under different scaling factors. When ω=2 and 4, all the deep learning-based methods except for the BP-ANN method achieve satisfactory performance. Among them, the proposed system slightly outperforms the other deep learning-based methods. It should be noted that all these deep learning-based methods need to re-train the model subjected to different super-resolutions; in contrast, the proposed method only needs to train the model one time and then can be applied to different super-resolutions with very good accuracy. The results also show that directly using the EDSR-2D method in time series load data does not achieve satisfactory performance. These 2D super-resolution methods usually have a large number of parameters to be trained, making them easy to be overfitted and not suitable for use in edge devices such as smart meters.

Table 5-1 also shows the proposed LSR system achieves better performance than the None-LRM method. Figure 5-3 further shows the training convergence (reflected in the MAE) of the proposed LSR method and the None-LRM method on both the training and validation datasets. The proposed system demonstrates superior convergence and achieves a lower MAE, highlighting the effectiveness of the LRM in the system. Figure 5-4 gives a specific example of applying different methods to reconstruct a 30-minute sampled load profile to a load profile with a resolution of 7.5 minutes. Generally, all the methods can well reconstruct the load profiles in such a small scaling factor (i.e., ω=4), and it can be seen that the proposed method performs better in terms of reconstructing the sudden transition load points.

TABLE 5-1. Comparison of load reconstruction performance under different methods (in kW)

LSR Method	$\omega = 2$ (30 to 15 min)			$\omega = 4$ (30 to 7.5 min)			$\omega = 15$ (30 to 2 min)			$\omega = 30$ (30 to 1 min)		
	MAE	*RMSE*	*FCE*	*MAE*	*RMSE*	*FCE*	*MAE*	*RMSE*	*FCE*	*MAE*	*RMSE*	*FCE*
Nearest interpolation	0.218	0.297	2.389	0.295	0.422	4.674	0.333	0.480	7.213	0.336	0.484	8.013
Cubic interpolation	0.233	0.316	2.540	0.296	0.413	4.442	0.332	0.466	6.346	0.335	0.470	6.755
SRPCNN method	0.190	**0.267**	**2.135**	0.252	0.368	4.018	N/A	N/A	N/A	N/A	N/A	N/A
EDSR-1D-based	0.198	0.275	2.192	0.258	0.372	4.053	N/A	N/A	N/A	N/A	N/A	N/A
EDSR-2D-based	N/A	N/A	N/A	0.264	0.378	4.151	N/A	N/A	N/A	N/A	N/A	N/A
BP-ANN-based	0.239	0.327	2.662	0.306	0.433	4.669	N/A	N/A	N/A	N/A	N/A	N/A
None-LRM	0.198	0.283	2.275	0.256	0.375	4.081	0.287	0.421	5.917	0.289	0.424	6.339
Proposed LSR	**0.188**	0.269	2.176	**0.247**	**0.362**	**3.969**	**0.278**	**0.409**	**5.849**	**0.280**	**0.412**	**6.297**

*"N/A" means "not applicable"; "min" means "minute".

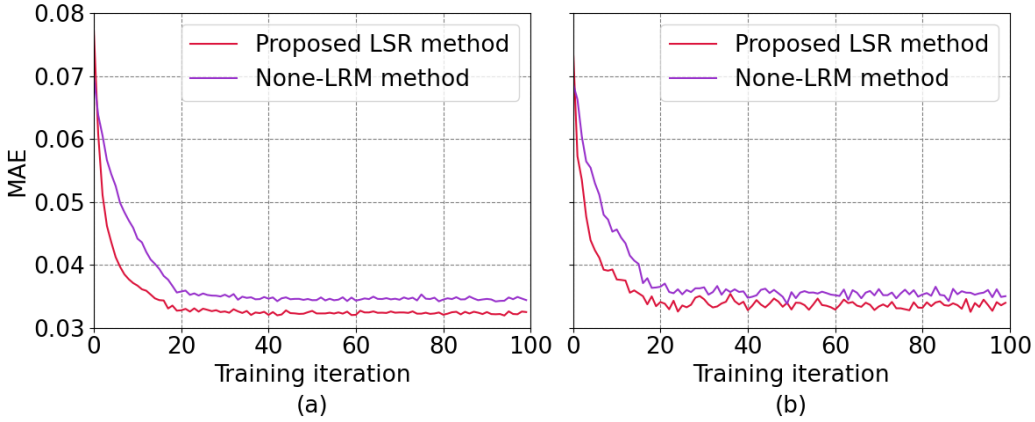


FIGURE 5-3. Training convergence comparison between the proposed method and the None-LRM method on (a) the validation dataset; (b) the testing dataset.

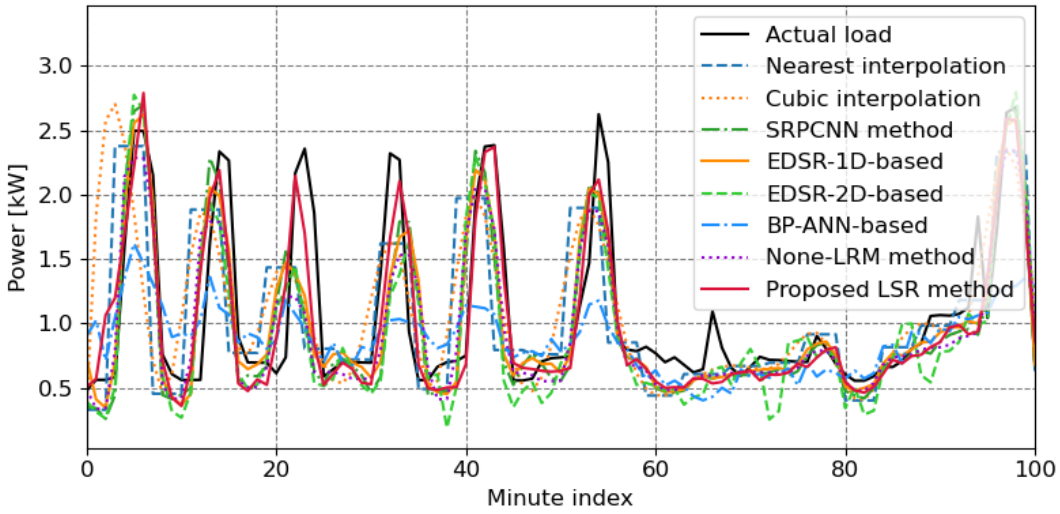


FIGURE 5-4. Comparison of load reconstruction results under the different methods in

case of ω=4.

A significant strength of the proposed technique is that it can reconstruct a low-resolution load profile into a load profile with an arbitrarily specified resolution, even if the resolution (i.e., the scaling factor) is never used in the training data. This cannot be achieved by the existing LSR methods. To validate this, Table 5-1 also reports the LSR performance generated by the different methods with scaling factors not used in model training, i.e., ω=15 and 30. The conventional deep learning-based methods are not applicable to these LSR tasks. The proposed method significantly outperforms the two interpolation methods. Figure 5-5 further gives an example of reconstructing a load profile from a 30- to a 2-minute resolution. The critical/sudden transitions points are computed by taking the first-order derivative of the load profile, which are shown in black markers in the figure. Overall, the results demonstrate the proposed method's effectiveness and its potential to flexibly support various upper-level energy applications.

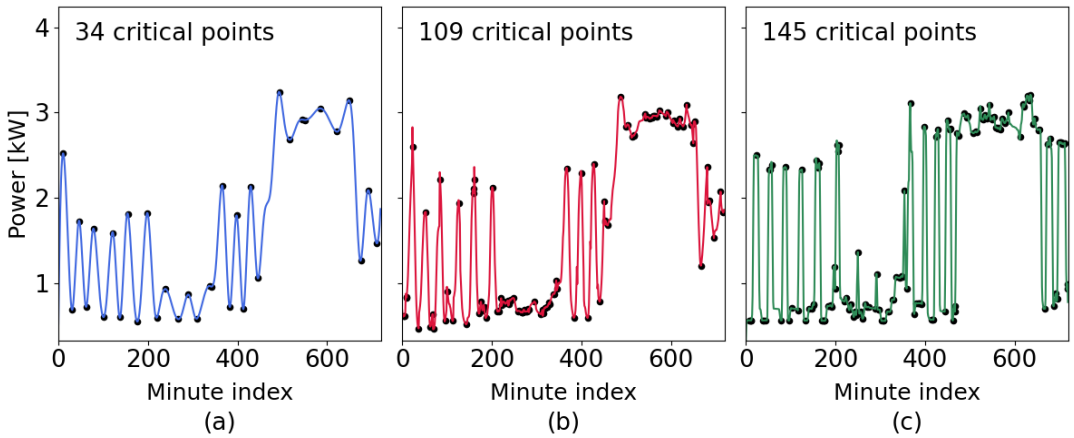


FIGURE 5-5. Comparison of load reconstruction and critical points with a scaling factor of 15 under (a) the cubic interpolation, (b) the proposed LSR method, and (c) the actual load profile.

5.4 Chapter Summary

This chapter presents a novel LSR framework capable of performing arbitrary-scale super-resolution, overcoming the limitations of conventional methods that are typically restricted to fixed scaling factors. The proposed approach trains a deep learning model

on low-resolution load data, enabling flexible reconstruction at any desired temporal granularity. This capability allows the framework to better accommodate the diverse requirements of real-world smart grid applications.

The proposed framework is built upon continuous representation learning, which introduces a coordinate-based system. Specifically, the model takes both the load values and their corresponding time coordinates as inputs during training, allowing it to learn a projection function that maps time coordinates to load values. During inference, by adjusting the input coordinates to a finer temporal resolution, the model can generate high-resolution load profiles from low-resolution data.

Comprehensive experiments on large-scale residential datasets demonstrated that the proposed method consistently outperforms interpolation-based and state-of-the-art fixed-scale LSR deep learning techniques.

CHAPTER 6

Meta-learning for STLF with Limited Training Data

This research study proposes a transferable model-agnostic meta-learning (T-MAML) approach for short-term load forecasting for single households. The proposed approach enables multiple households to collaboratively train a generic artificial neural network (ANN) model. The generic ANN model is then further trained at each target household node for STLF purposes. The proposed T-MAML-based STLF approach is featured by: (1) significant reduction of computation and communication costs on the household side; and (2) superior STLF performance, especially when there is limited load data for training in a target household. Experiments based on a real Australian residential dataset are conducted to validate the effectiveness of the proposed approach.

6.1 Introduction

Short-Term Load Forecasting (STLF) plays a fundamental role in modern energy systems. While traditional STLF is performed at the bus level, the widespread deployment of advanced metering infrastructure has increased the development work of STLF for individual households.

Machine learning techniques have been applied for STLF of single households, e.g., support vector machine [175], random forest regressor [176], and deep learning [177]. Recently, the model-agnostic meta-learning (MAML) has been applied to household STLF [178], which uses a household's historical load data before noon to train an MAML model, and then uses the model to predict its load data in afternoon hours. While the work [4] demonstrates the effectiveness of MAML in STLF, as in many other literatures (e.g., [175-177]), it assumes there is sufficient data to train the model. In other words, the work [175-178] does not deal with the situation when the household load data is limited for training – this is a common case in various scenarios (e.g., short-

term accommodation and newly moved-in households). In such a limited training data situation, the accuracy of STLF would be significantly degraded. Federated Learning (FL)-based method [179] can help to alleviate this issue. It enables multiple households to collaboratively update the parameters of their local models without exchanging the data; in this way, different households can "help each other" to train their own models. However, the FL process requires compute-intensive training on the household side and repeated communications between the households and a central node; these pose a non-trivial demand on the underlying information infrastructure in the residential environment.

This research study proposes a Transferrable MAML (T-MAML) based STLF system tailored for the single household environment with limited training data and limited computing power. Firstly, the system performs a "*meta-training*" process, in which the system enables multiple households with accessible load data to collaboratively train a generic learning model that is with well-trained parameters and can be easily customized for different target households. The system then applies a fine-tuning procedure to customize the parameters of the generic model based on the individual target household's data – this process is referred to as a "*meta-transfer*". The tuned model on each household side is then used for STLF. Compared with the FL-based methods, the proposed system does not require repeated communications nor much computation on the household side; it only needs a small amount of data from a target household to generate the final ANN model for the STLF. With these features, the system is highly suitable for use in residential environments. Compared with the MAML-based STLF system in [178], which only consists of the meta-training process, the proposed system integrates both the meta-training and the meta-transfer processes so that the system can not only well represent the generic features of a group of households but also can effectively capture the specific features of a target household. As a result, as shown in the experiment, the T-MAML-based approach can achieve superior performance.

In the rest of this chapter, we present the design principles of the proposed system

(Section 6.2), report the experiments (Section 6.3), and provide concluding remarks (Section 6.4).

6.2 T-MAML Approach for Single Household STLF

The single household STLF scenario can be depicted as Figure 6-1, which includes a set of households with accessible load data (denoted as "sources") and a set of households that are the targets of the STLF. For each household h (either a source or a target), its historically recorded load data is expressed as:

$$\mathbf{P}_h = [P_{h,1}, ..., P_{h,R_h}] \tag{6.1}$$

where R_h is the number of load records of household h and each record $P_{h,r}$ (r=1: R_h) denotes the power demand of the household h at a particular time interval (kW). The system then performs STLF following two steps: a "meta-training" step and a "meta-transfer" step.

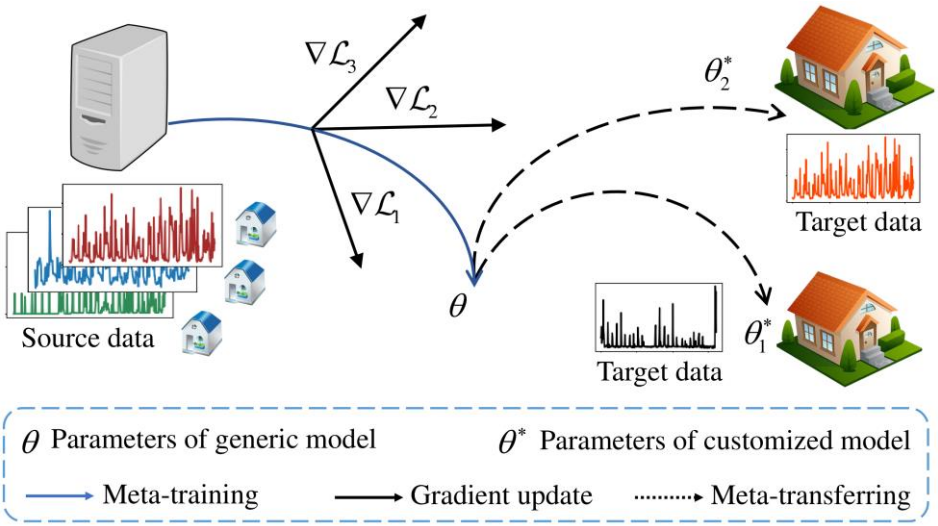


FIGURE 6-1. Schematic of the T-MAML based STLF system for single households.

6.2.1 Meta-Training Process

The meta-training process is performed on a central node. The system gathers multiple sources' data to train a generic ANN model that is parameterized by a function

$f_{\boldsymbol{\theta}}$ (θ is the parameter vector) such that the parameters can be effectively further trained subjected to each target's own data.

The meta-training process is performed in an iterative manner. At the beginning, the system initializes a generic model with random parameters; then, in each iteration, the system randomly selects a certain number of sources and applies a learning process with two nested loops: an inner loop and an outer loop. In the inner loop, the system generates a set of temporary parameters based on the data of each randomly selected source. In the outer loop, the system updates the parameters of the generic model based on the gradients of the temporary parameters of the sources. Such a nested learning process reflects the so-called "learning-to-learning" philosophy of MAML [180], enabling the outer loop learning to balance the inner loop learning on different sources and finally achieving a generic model that can be easily further customized in the different targets.

More specifically, denote the set of sources as Φ^{S}, the system randomly selects M sources from Φ^{S} to generate a subset $\Phi^{S',(n)}$ ($\Phi^{S',(n)} \in \Phi^{S}$ and $\left|\Phi^{S',(n)}\right| = M$) in the nth round of the iteration (n=1:N). For each source $m \in \Phi^{S'(n)}$, K pairs of samples are randomly selected as a support set $\Phi_m^{sup,(n)}$ and a query set $\Phi_m^{que,(n)}$, respectively. In the inner loop, $f_{\boldsymbol{\theta}}$ with model parameters θ is trained on $\Phi_m^{sup,(n)}$ by using J steps of Stochastic Gradient Descent (SGD):

$$\hat{\theta}_m^{(n)} = \theta^{(n)} - \alpha \nabla_{\theta^{(n)}} \mathcal{L}_{\Phi_m^{sup,(n)}}(f_{\theta^{(n)}}),\ m \in \Phi^{S',(n)}, n = 1:N \tag{6.2}$$

where α is the inner learning rate; $\hat{\theta}_m^{(n)}$ is the updated parameters of $f_{\boldsymbol{\theta}}$ on $\Phi_m^{sup,(n)}$ in the nth round; $\mathcal{L}(\cdot)$ denotes the loss function (Eq. (6.3)), and the subscript ($\Phi_m^{sup,(n)}$ or $\Phi_m^{que,(n)}$) represents the dataset to which it applies:

$$\mathcal{L}_{\Phi_m^{sup,(n)}}(f_{\theta^{(n)}}) = \sum_{\mathbf{x},\mathbf{y} \sim \Phi_m^{sup,(n)}} \left\| f_{\theta^{(n)}}(\mathbf{x}) - \mathbf{y} \right\|_2^2 \tag{6.3}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the "input-output" pair, where the input is a vector of sequential

load values and the output is the actual load value. The model outputs a predicted load (denoted as $f_{\theta^{(n)}}(\mathbf{x})$), which is then compared with the actual load in the support set $\Phi_m^{sup,(n)}$. The application of Eq. (6.2) produces M temporary models with respect to M sources, where each temporary model $f_{\hat{\theta}_m^{(n)}}$ ($m \in \Phi^{S',(n)}$) is represented by the parameters $\hat{\theta}_m^{(n)}$. Based on the temporary models, the system computes the gradients and updates the model parameters with respect to the generic model θ, aiming at minimizing the total loss on the query sets $\Phi_m^{que,(n)}$ of all the sources in $\Phi^{S',(n)}$:

$$\begin{aligned} F &= \arg\min_{\theta} \sum_{m \in \Phi^{S',(n)}} \mathcal{L}_{\Phi_m^{que,(n)}}(f_{\hat{\theta}_m}) \\ &= \arg\min_{\theta} \sum_{m \in \Phi^{S',(n)}} \mathcal{L}_{\Phi_m^{que,(n)}}(f_{\theta} - \alpha \nabla_{\theta} \mathcal{L}_{\Phi_m^{sup,(n)}}(f_{\theta})) \end{aligned} \tag{6.4}$$

where $\theta^{(n)}$ is updated with the application of SGD:

$$\theta^{(n+1)} \leftarrow \theta^{(n)} - \beta \sum_{m \in \Phi^{S',(n)}} \nabla_{\theta^{(n)}} \mathcal{L}_{\Phi_m^{que,(n)}}(f_{\hat{\theta}_m^{(n)}}) \tag{6.5}$$

where β is the outer learning rate. In this study, we design a validation process for the generic model θ, which is performed on a set of validation sources Φ^V ($\Phi^V \cap \Phi^S = \varnothing$ and $|\Phi^V| < |\Phi^S|$). In each iteration, the data of each validation source $v \in \Phi^V$ is divided into a support set and a query set. θ is firstly trained on the support set with J steps of SGD and then is validated on the query set. The performance of θ is then evaluated in terms of the average validation loss computed across all the validation sources.

The above parameters update process for θ iteratively proceeds until a maximum iteration number N is reached. The generic model with the lowest validation loss in the iteration is used as the output (f_{θ}). The outputted model will be further customized for each target with a meta-transfer process.

6.2.2 Meta-Transfer Process

After the meta-training process, the generic model f_θ is broadcasted to the targets to launch the meta-transfer process. This is achieved by using each target household's load data to further tune the model f_θ separately (also known as "fine-tuning"). For a specific target household *a*, the parameters of the ANN model generated from the meta-training process are customized with only a small amount of historical energy load data from the target. Assume θ is the generic model received from the central node, and the system iteratively updates θ using the target's load data:

$$\theta^{(w+1)} \leftarrow \theta^{(w)} - \varepsilon \nabla_{\theta^{(w)}} \mathcal{L}_{D_a}(f_{\theta^{(w)}}) \tag{6.6}$$

where ε is the learning rate of fine-tuning; w is the iteration index of SGD; D_a is the load dataset owned by the target *a*. The iteration process in Eq. (6.6) terminates when a pre-specified maximum iteration number W is reached.

As shown in Section 6.3, since the meta-training process already offers a well-tuned generic model, setting W to be a small number could yield a model with good performance. This means that the meta-transfer process only needs to utilize a small amount of target load data, making the system very computationally efficient on the household side.

6.3 Experiment

6.3.1 Experiment Setup

We extract 100 households from the Australian "Smart Grid, Smart City (SGSC)" dataset [150]. The dataset records power consumption data of more than 9,000 residents in the Great Sydney area, spanning from 2010 to 2014. 30 households are selected as sources and 60 are used as targets. 10 households are selected to form the validation set in the meta-training process. The time resolution of the load data is 1 hour. Following

references [177] and [179], the models' input and output pairs are generated by the sliding window method with the lookback and lookahead sizes set as 12 and 1.

The system is model-agnostic, meaning that it can incorporate any gradient-based learning model. A 3-layer Fully Connected Network (FCN) and a Convolutional Neural Network (CNN) are used in this simulation. The values of the parameters K, J, M, and W are set to be 12, 5, 10, and 5, respectively. The values of the learning rates α, β, and ε are set to be 0.001, 0.1, and 0.1, respectively. All the programs are implemented in Python and are executed on a computer with an AMD Ryzen 5 3500X 6-core processor and an NVIDIA GeForce GTX 3060 GPU.

6.3.2 Results and Discussion

We compare several methods: (i) the aggregated learning (AL)-based methods - the data of all sources are collected to train a model first, and then the fine-tuning is applied to generate the STLF model for each target. These methods also represent the basic transfer learning principles. (ii) A MAML method, which directly uses the generic model generated in meta-training for STLF for the targets. This represents the application of the approach in [4] on multiple source households. (iii) The proposed T-MAML-based STLF system; (iv) the FL-based method [179]. And (v) 4 conventional STLF methods, including a *k*-Nearest Neighbor (KNN) regression method, a Linear Regression (LR) method, a Random Forest (RF) regressor, and a Ridge regressor. These conventional methods only use the target's data for training and forecasting. The Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE) are used as evaluation metrics.

Table 6-1 reports the STLF results generated by the different methods for the 70 targets under 3 limited training data scenarios, in which each target is assigned with only 1-day, 3-day, and 7-day load data, respectively. Clearly, with the reduction of the available data amount (i.e., from 7-, 3-, to 1-day data), while the performance of all the methods decreases, the T-MAML based approaches achieve superior performance to

the others. For example, when the targets have 7-day energy load data for training, the T-MAML+CNN method achieves a lower MAPE (by 4.73%) than the one obtained with the AL+CNN method, while it reaches a better performance (of 14.89%) when the targets have only 1-day load data. In particular, the experiment has demonstrated the effectiveness of the model customization process, i.e., the meta-transfer process. As can be seen in Table 6-1, by applying only 5 steps of fine-tuning, the proposed system (with either FCN or CNN as the learning model) yields significantly superior performance than the MAML approaches without meta-transfer; this is evidenced by the large reduction of both the MAPE and RMSE values.

TABLE 6-1. STLF performance comparison

	1-day data		3-day data		7-day data	
	MAPE	*RMSE*	*MAPE*	*RMSE*	*MAPE*	*RMSE*
T-MAML+CNN	73.39	0.200	68.35	0.198	66.09	0.197
T-MAML+FCN	71.20	0.201	67.08	0.198	65.02	0.198
MAML+CNN	247.25	0.340	247.25	0.340	247.25	0.340
MAML+FCN	119.72	0.246	119.72	0.246	119.72	0.246
AL+CNN	88.28	0.204	75.10	0.197	70.82	0.196
AL+FCN	91.88	0.204	79.12	0.199	75.39	0.198
FL+CNN	125.26	0.220	115.84	0.213	113.81	0.209
FL+FCN	119.19	0.212	110.99	0.208	105.18	0.208
RF	168.96	0.289	90.62	0.222	81.14	0.209
KNN	148.38	0.271	94.66	0.230	82.81	0.219
LR	1901.62	3.686	97.01	0.218	81.94	0.198
Ridge	151.69	0.279	92.73	0.208	81.09	0.196

*"X+CNN" and "X+FCN" (X stands for MAML, FL, or AT) mean using a CNN or FCN as the learning model in the T-MAML, MAML, FL framework, or AL strategy, respectively. The MAPE and RMSE values under the "MAML+CNN" and "MAML+FCN" approaches under different scenarios are identical because the target-side load data is not used in these approaches.

We investigate the impact of different values of the maximum iteration number in the meta-transfer process (W) on the system's performance. Table 6-2 reports the comparison result of the performance of the T-MAML+CNN approach under the different settings of W. It can be seen that when the target dataset is of a very small size (i.e., the 3-day data case), over updating the model's parameters would lead to

overfitting, which subsequentially degrades the system's performance (reflected in the increased MAPE values).

We further compare the computational efficiency between the T-MAML- and FL-based approaches for STLF. Table 6-3 reports the comparison result under a scenario in which each household has 7-day data for training. The FL-based approach is more computationally efficient in terms of the total model training time; this is because it does not include a centralized meta-training-like process. In the FL-based approach, all households train the local models in parallel in one iteration. In the proposed system, the meta-training step consumes most of the computation time as it aggregately trains the data from the sources. In this simulation, we use 22-day data from the sources for training; as a result, the total model training time under the T-MAML-based method is 750.8 seconds. Despite the T-MAML-based method spending more time on meta-training, it significantly reduces the computational cost on the household side. It only takes 0.04 seconds to tune the generic model generated in meta-training and make the model ready for STLF at the target household nodes; in contrast, all the computation tasks in the FL-based method are undertaken by the households. Since the meta-training step can be performed offline, the result indicates that the T-MAML based approach can produce the real-time STLF result very efficiently (i.e., only uses 0.04 seconds on the household side). In addition, there are a total of 6,000 instances of communication between the households and the central node needed in the whole FL process, while that number in the T-MAML-based method is only 60 (required for the households to receive the parameters of the generic model). Again, this highlights that the proposed method has significantly less requirement on the underlying communication infrastructure than the FL-based methods.

TABLE 6-2. STLF performance with different settings of W in meta-transfer

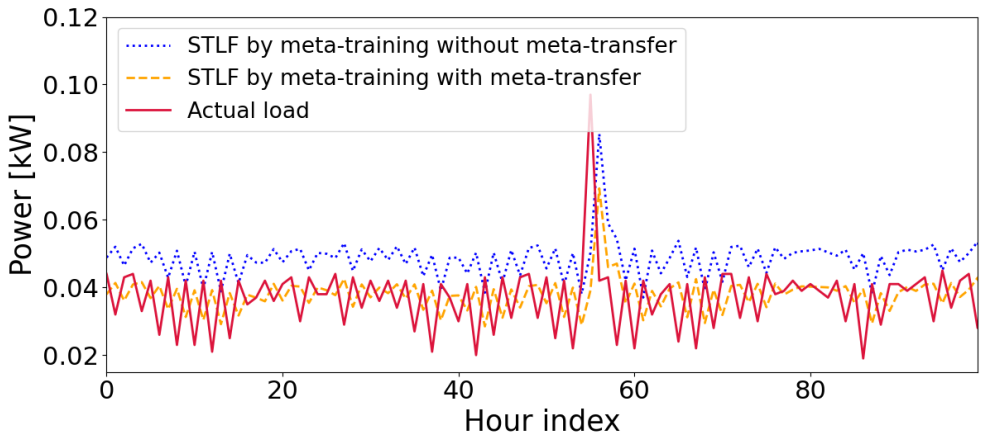
	3-day data		7-day data	
	MAPE	*RMSE*	*MAPE*	*RMSE*
W=5	68.35	0.198	66.09	0.197
W=20	72.27	0.197	70.22	0.196
W=50	73.18	0.196	70.57	0.195

TABLE 6-3. Computational Efficiency between MAML- and FL-based methods for STLF under a 7-day training data scenario

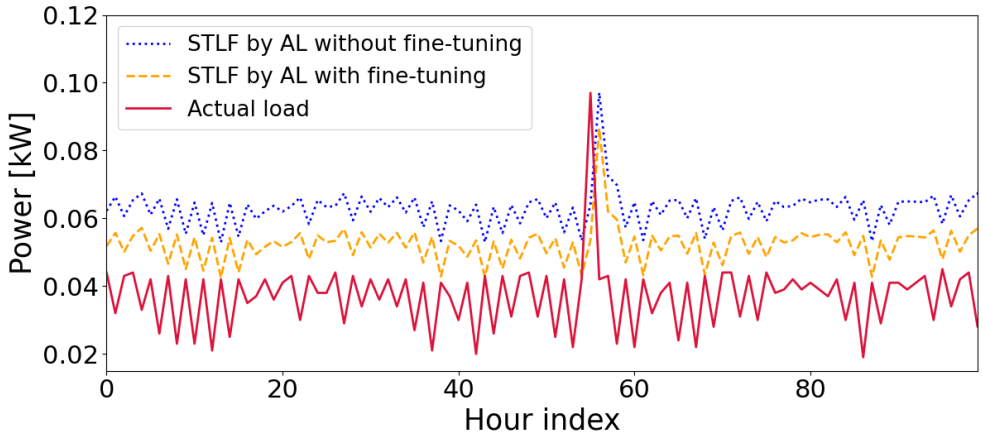
	Total model training time	Average model training time in a household	Total number of communications
FL+ANN	65.8s	65.8s	6,000
MAML+ANN	750.8s	0.04s	60

Figure 6-2 further demonstrates the effectiveness of the meta-training mechanism in the proposed system through the comparison with an AL-based approach on a randomly selected target household. Both are with the FCN as the learning models and are with the same fine-tuning process. As indicated by the red lines in Figure 6-2, the STLF results obtained with the generic model generated by meta-training can better follow the actual load profile than the AL-based approach. Using this well-trained generic model as the starting point, the system uses only 5 iterations to further tune it to achieve satisfactory STLF performance for the target, as indicated by the blue dotted lines.

Figure 6-3 shows the comparison among the proposed system, the FL-based methods and the AL-based methods on the MAPE values across all 60 targets. For most of the targets, the proposed system leads to smaller MAPE values than the others. Figure 6-4 further shows the comparison among the different methods on a single target, with the FCN used as the learning model. When the target's training data is insufficient, the conventional approach (i.e., FCN), which directly trains a model based on the target's data, fails to achieve reasonable prediction. Among the collaborative learning-based methods, the T-MAML-based approach exhibits the best performance.



(a)



(b)

FIGURE 6-2. Comparison of MAML- (a) and AL-based (b) approaches on a target household.

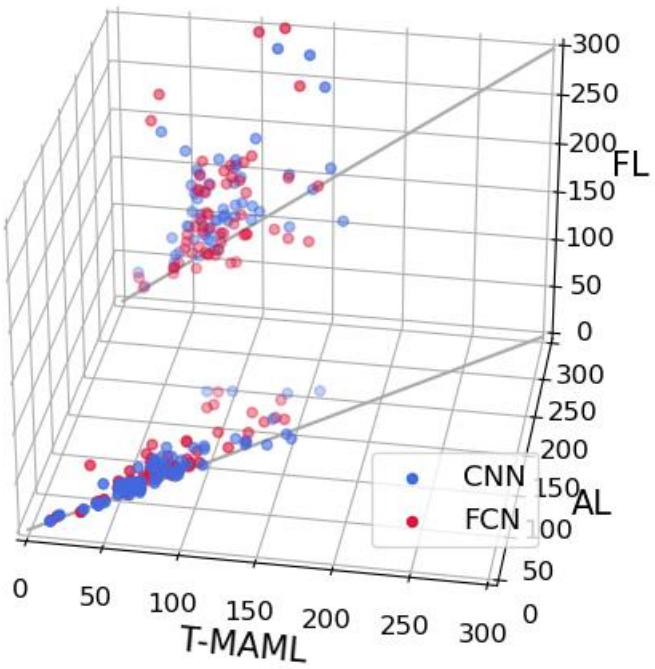


FIGURE 6-3. Comparison of MAPE under different methods across 60 targets.

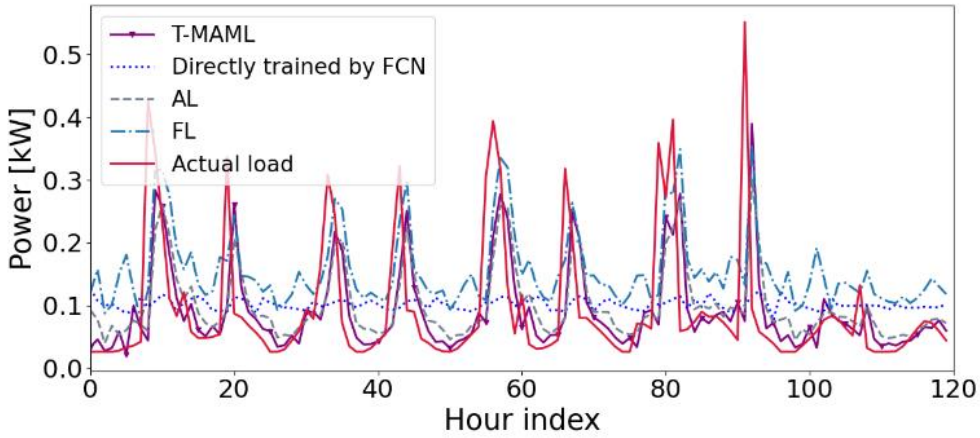


FIGURE 6-4. STLF result comparison on a single target household (ID: 10100412).

6.4 Chapter Summary

This chapter introduces a meta-learning-based framework for the STLF task. This work aims to address the challenge of enabling users in newly occupied households and small-scale residential environments to train their customized local models. Unlike conventional models that require extensive user-specific training data, the proposed

approach leverages a meta-learning strategy to enable rapid adaptation across heterogeneous users.

The framework comprises two central stages. First, a meta-training phase is conducted over a diverse set of households to train and obtain a generic forecasting model with well-initialized parameters. This generic model serves as a transferable prior, encapsulating knowledge of common load dynamics across users. Second, a meta-transfer phase fine-tunes the model to new households with limited data, ensuring personalized forecasting without heavy computational or communication burdens. This design allows the system to retain strong generalization capabilities while delivering accurate predictions in data-scarce scenarios.

Experimental evaluations on real-world smart meter datasets validate the effectiveness of the proposed meta-learning framework, showing superior performance over federated learning and individually-user trained baselines in terms of both forecasting accuracy and training efficiency. Remarkably, the framework can adapt to user-specific conditions using as little as three days of hourly-resolution data and achieves this with significantly reduced training time compared to federated learning frameworks, making it highly practical for real-world residential applications.

CHAPTER 7

Conclusions and Future Directions

7.1 Conclusions

This thesis aims to develop deployment-friendly frameworks for STLF and LSR techniques within modern energy systems. The major contribution of this thesis can be summarized as follows.

The first research study (see Chapter 3) proposes a hierarchically federated framework for facilitating collaborative short-term load forecasting for residential energy users. The framework first uses a privacy-preserving user clustering method to divide the users into multiple clusters based on their load similarity; then, a hierarchically federated learning-based mechanism is applied to train the STLF model through the collaboration among the three types of nodes in the system, i.e., a global node, multiple cluster heads, and the users.

The developed framework is validated on a real Australian residential load dataset. The experiment results have demonstrated several merits of the framework compared with the existing STLF methods: (i) high load forecasting accuracy. By establishing a unique, hierarchically federated mechanism (i.e., intra- and inter-cluster FL processes), the system can exploit the latent information in all the users' load datasets for load forecasting while ensuring that major knowledge exchange is performed among the users with similar load patterns. As a result, this significantly improves the STLF accuracy; (ii) high private-preserving capability. By combining the federated user clustering method and the hierarchically federated model training mechanism, the load data of the users do not have to be revealed in the whole collaborative STLF process; this fully ensures the users' data privacy; and (iii) high fault tolerance. The asynchronous communication mechanism in the system makes it well-adapted to communication uncertainties in the residential environment.

The second research study shown in Chapter 4 proposes an unsupervised LSR framework that represents a significant step forward in power load data reconstruction. It can eliminate the need to use both high- and low-resolution load data for training the LSR model, enabling load reconstruction solely from low-resolution load data that could be sampled from aging smart meters with limited sampling resolution – this makes the system highly suitable for real-world applications. Such an ability of the system is achieved by integrating contrastive and internal learning mechanisms to analyze the relationship between the load data with a low resolution and a further lower resolution, which inherently exists in the input load data. Besides, the system can handle diverse noise signals in practical power load sampling processes well.

The proposed unsupervised LSR system is evaluated subjected to a variety of metrics and is compared with conventional supervised LSR methods. The results show that the proposed method vastly outperforms the other methods when considering noise. In the situation where noises are not considered, the proposed method still achieves a comparable performance with the supervised learning-based LSR methods – considering the proposed method performs LSR in an unsupervised way and does not have high-resolution load data for ex-ante training, this result is very encouraging. Meanwhile, the experiments show the proposed method performs satisfactorily and stably in different LSR tasks with different scaling factor settings. Overall, the experiment results suggest promising potential of the proposed system in real-world applications.

The research study in Chapter 5 reports on a new LSR technique. Through constructing a 1-dimensional coordinate system and mapping training load data to it, the technique can generate load values at any coordinates in the coordinate system and thus can reconstruct a low sampling resolution load profile to be a load profile with an arbitrarily specified high resolution that could represent the actual variation of the load. Experiments on real-world load data are conducted to validate its effectiveness.

The research study in Chapter 6 reports on a transferable model-agnostic meta-learning approach for STLF. The method uses multiple households' knowledge to train

a generic learning model, and it then applies a fine-tuning process to customize the model to specific target households. Compared with other collaborative learning frameworks, the proposed approach is characterized by significantly less communication and computation cost, and it is suitable for application on targets with limited data for training. Its superior performance is validated based on an Australian residential load dataset.

7.2 Future Directions

Beyond the challenges addressed in this thesis, future research could be pursued along several directions, including but not limited to the following.

The first research study shown in Chapter 3 can be conducted in several directions. Encryption technologies can be applied to the developed framework to ensure that the exchanged model parameters cannot be eavesdropped on by attackers, which would further enhance the system's security. Besides, social networks can be modeled and integrated into the developed framework to achieve social relationship-aware STLF collaboration among the users, and this is the author's current work.

In terms of the second research study shown in Chapter 4, future research can be built upon the work presented in this study. While the experiments show the proposed LSR technology can well adapt to input load data with relatively low sampling resolutions (i.e., every 15 minutes and every 5 minutes, see Section 4.6), our lab trials (not reported in this study) also indicate that for 1-day LSR tasks (that is, tasks aiming to reconstruct load on daily basis), effective LSR can hardly be achieved by artificial neural network-based methods (including the proposed method) when the original input load data is with a very low sampling resolution (e.g., one hour). This is because when the resolution of the input load data is too low, each 1-day load data profile will only contain a small number of load values, making the neural network model hardly trained. In the future, it is worth investigating unsupervised LSR techniques targeting very low-resolution load data. Besides, it is also worth evaluating the effectiveness of the

technology presented in this study in terms of recovering load data from minute-to-second-level resolutions.

In terms of the research study in Chapter 5, one potential limitation of the LSR system presented in this chapter is that, like other LSR methods, the model needs to be well-trained to achieve satisfactory performance. When the training data is not sufficient, the system will not perform well. Currently, the authors are developing a federated training framework to enable multiple nodes to jointly train the LSR model based on their small datasets.

In terms of the research study in Chapter 6, future work can be conducted in different directions. Currently, the authors are working on developing privacy-preserving mechanisms for the T-MAML-based STLF system to enable meta-training without explicitly exposing the sources' data. In addition, it is worth investigating how the load pattern diversity among the source households and target households would affect the load forecasting accuracy; proper strategies (e.g., integrating load clustering into T-MAML) can be developed to reduce the negative impact of such diversity.

Bibliography

[1] A. K. Chakraborty and N. Sharma, "Advanced metering infrastructure: Technology and challenges," in *Proc. IEEE/PES transmission and distribution conference and exposition*, 2016, pp. 1-5.

[2] S. K. Rathor and D. Saxena, "Energy management system for smart grid: An overview and key issues," *International Journal of Energy Research,* vol. 44, no. 6, pp. 4067-4109, 2020.

[3] K. Kok, "Dynamic pricing as control mechanism," in *Proc. IEEE Power and Energy Society General Meeting*, 2011, pp. 1-8.

[4] S. Mansouri, F. Zishan, O. D. Montoya, M. Azimizadeh, and D. A. Giral-Ramírez, "Using an intelligent method for microgrid generation and operation planning while considering load uncertainty," *Results in Engineering,* vol. 17, p. 100978, 2023.

[5] Q. Wang, C. Zhang, Y. Ding, G. Xydis, J. Wang, and J. Østergaard, "Review of real-time electricity markets for integrating distributed energy resources and demand response," *Applied Energy,* vol. 138, pp. 695-706, 2015.

[6] D. Patteeuw, K. Bruninx, A. Arteconi, E. Delarue, W. D'haeseleer, and L. Helsen, "Integrated modeling of active demand response with electric heating systems coupled to thermal energy storage systems," *Applied Energy,* vol. 151, pp. 306-319, 2015.

[7] E. O'driscoll and G. E. O'donnell, "Industrial power and energy metering–a state-of-the-art review," *Journal of cleaner production,* vol. 41, pp. 53-64, 2013.

[8] M. M. Forootan, I. Larki, R. Zahedi, and A. Ahmadi, "Machine learning and deep learning in energy systems: A review," *Sustainability,* vol. 14, no. 8, p. 4832, 2022.

[9] A. O. Aseeri, "Effective RNN-based forecasting methodology design for improving short-term power load forecasts: Application to large-scale power-

grid time series," *Journal of Computational Science,* vol. 68, p. 101984, 2023.

[10] Y. Eren and İ. Küçükdemiral, "A comprehensive review on deep learning approaches for short-term load forecasting," *Renewable and Sustainable Energy Reviews,* vol. 189, p. 114031, 2024.

[11] Y. He, F. Luo, and G. Ranzi, "Load reconstruction with arbitrary super resolutions," *IEEE Transactions on Power Systems,* pp. 1-4, 2023.

[12] C. Zhang, Z. Shao, and F. Chen, "A power data reconstruction method based on super-resolution generative adversarial network," in *Proc. Asia Conference on Power and Electrical Engineering*, 2021, pp. 300-304.

[13] X. Ying, "An overview of overfitting and its solutions," in *Proc. Journal of Physics*, 2019, p. 022022.

[14] Q. Dong *et al.*, "Short-term electricity-load forecasting by deep learning: A comprehensive survey," *Engineering Applications of Artificial Intelligence,* vol. 154, p. 110980, 2025.

[15] H. Teiwes, S. Blume, C. Herrmann, M. Rössinger, and S. Thiede, "Energy load profile analysis on machine level," *Procedia Cirp,* vol. 69, pp. 271-276, 2018.

[16] G. Box and G. Jenkins, "Analysis: Forecasting and Control," *San francisco,* 1976.

[17] G. E. Box and G. M. Jenkins, "Some recent advances in forecasting and control," *Journal of the Royal Statistical Society. Series C (Applied Statistics),* vol. 17, no. 2, pp. 91-109, 1968.

[18] B. Dhaval and A. Deshpande, "Short-term load forecasting with using multiple linear regression," *International Journal of Electrical and Computer Engineering,* vol. 10, no. 4, pp. 3911-3917, 2020.

[19] S. Kyung-Bin, B. Young-Sik, H. Dug Hun, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," *IEEE Transactions on Power Systems,* vol. 20, no. 1, pp. 96-101, 2005.

[20] A. Y. Saber and A. K. M. R. Alam, "Short term load forecasting using multiple linear regression for big data," in *Proc. 2017 IEEE Symposium Series on*

Computational Intelligence, 2017, pp. 1-6.

[21] L. Shang, K. Chen, G. Wang, Y. Liu, R. Hu, and Y. Shang, "Short-term distribution network peak load forecasting based on generalized linear model," in *Proc. 2022 4th International Conference on Power and Energy Technology*, 2022, pp. 584-589.

[22] N. Amral, C. S. Ozveren, and D. King, "Short term load forecasting using Multiple Linear Regression," in *Proc. 42nd International Universities Power Engineering Conference*, 2007, pp. 1192-1198.

[23] G. Dudek, "Pattern-based local linear regression models for short-term load forecasting," *Electric Power Systems Research,* vol. 130, pp. 139-147, 2016.

[24] N. Mohamed, M. H. Ahmad, Z. Ismail, and S. Suhartono, "Short term load forecasting using double seasonal arima model," in *Proc. 2021 Regional Conferance on Statistical Sciences*, 2010, vol. 10, pp. 57-73.

[25] M. Y. Cho, J. C. Hwang, and C. S. Chen, "Customer short term load forecasting by using ARIMA transfer function model," in *Proc. 1995 International Conference on Energy Management and Power Delivery*, 1995, vol. 1, pp. 317-322 vol.1.

[26] C.-M. Lee and C.-N. Ko, "Short-term load forecasting using lifting scheme and ARIMA models," *Expert Systems with Applications,* vol. 38, no. 5, pp. 5902-5911, 2011.

[27] L. C. M. de Andrade and I. N. da Silva, "Very short-term load forecasting based on ARIMA model and intelligent systems," in *Proc. 15th International Conference on Intelligent System Applications to Power Systems*, pp. 1-6.

[28] D. Alberg and M. Last, "Short-term load forecasting in smart meters with sliding window-based ARIMA algorithms," *Vietnam Journal of Computer Science,* vol. 5, no. 3, pp. 241-249, 2018.

[29] M. A. Al Amin and M. A. Hoque, "Comparison of ARIMA and SVM for short-term load forecasting," in *Proc. 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference*, 2019, pp. 1-

6.

[30] S. Karthika, V. Margaret, and K. Balaraman, "Hybrid short term load forecasting using ARIMA-SVM," in *Proc. 2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2017, pp. 1-7.

[31] H. Nie, G. Liu, X. Liu, and Y. Wang, "Hybrid of ARIMA and SVMs for short-term load forecasting," *Energy Procedia,* vol. 16, pp. 1455-1460, 2012.

[32] S. Chen, R. Lin, and W. Zeng, "Short-term load forecasting method based on ARIMA and LSTM," in *Proc. 2022 IEEE 22nd International Conference on Communication Technology*, 2022, pp. 1913-1917.

[33] L. Tang, Y. Yi, and Y. Peng, "An ensemble deep learning model for short-term load forecasting based on ARIMA and LSTM," in *Proc. 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids*, 2019, pp. 1-6.

[34] N. Pooniwala and R. Sutar, "Forecasting short-term electric load with a hybrid of ARIMA model and LSTM network," in *Proc. 2021 International Conference on Computer Communication and Informatics*, 2021, pp. 1-6.

[35] C. Tarmanini, N. Sarma, C. Gezegin, and O. Ozgonenel, "Short term load forecasting based on ARIMA and ANN approaches," *Energy Reports,* vol. 9, pp. 550-557, 2023.

[36] L. Jian-Chang, N. Dong-Xiao, and J. Zheng-Yuan, "A study of short-term load forecasting based on ARIMA-ANN," in *Proc. 2004 International Conference on Machine Learning and Cybernetics*, 2004, vol. 5, pp. 3183-3187.

[37] I. Zuleta-Elles, A. Bautista-Lopez, M. J. Cataño-Valderrama, L. G. Marín, G. Jiménez-Estévez, and P. Mendoza-Araya, "Load forecasting for different prediction horizons using ANN and ARIMA models," in *Proc. 2021 IEEE Conference on Electrical, Electronics Engineering, Information and Communication Technologies*, 6-9 Dec. 2021 2021, pp. 1-7, doi: 10.1109/CHILECON54041.2021.9702913.

[38] P. Ji, D. Xiong, P. Wang, and J. Chen, "A study on exponential smoothing model

for load forecasting," in *Proc. 2012 Asia-Pacific Power and Energy Engineering Conference*, 2012, pp. 1-4.

[39] W. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems,* no. 2, pp. 900-911, 2007.

[40] L. Abderrezak, M. Mourad, and D. Djalel, "Very short-term electricity demand forecasting using adaptive exponential smoothing methods," in *Proc. 15th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, 2014, pp. 553-557.

[41] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE Transactions on Power Systems,* vol. 27, pp. 458-464, 2012.

[42] R. C. Souza, M. Barros, and C. V. C. Miranda, "Short term load forecasting using double seasonal exponential smoothing and interventions to account for holidays and temperature effects," *TLAIO II-2 do Taller Latino Iberoamericano de Investigación de Operaciones. Acapulco, México,* pp. 1-8, 2007.

[43] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE transactions on Power Systems,* vol. 27, no. 1, pp. 458-464, 2011.

[44] S. A. Karim and S. A. Alwi, "Electricity load forecasting in UTP using moving averages and exponential smoothing techniques," *Applied Mathematical Sciences,* vol. 7, no. 77-80, pp. 4003-4014, 2013.

[45] W. Sulandari and H. Utami, "Forecasting electricity load demand using hybrid exponential smoothing-artificial neural network model," *International Journal of Advances in Intelligent Informatics,* vol. 2, no. 3, pp. 131-139, 2016.

[46] J. Mohammed, S. Bahadoorsingh, N. Ramsamooj, and C. Sharma, "Performance of exponential smoothing, a neural network and a hybrid algorithm to the short term load forecasting of batch and continuous loads," in *Proc. 2017 IEEE Manchester PowerTech*, 2017, pp. 1-6.

[47] G. Dudek, P. Pełka, and S. Smyl, "A hybrid residual dilated LSTM and

exponential smoothing model for midterm electric load forecasting," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 33, no. 7, pp. 2879-2891, 2021.

[48] S. Smyl, G. Dudek, and P. Pełka, "ES-dRNN: A hybrid exponential smoothing and dilated recurrent neural network model for short-term load forecasting," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 35, no. 8, pp. 11346-11358, 2023.

[49] S. Smyl, G. Dudek, and P. Pełka, "Contextually enhanced ES-dRNN with dynamic attention for short-term load forecasting," *Neural Networks,* vol. 169, pp. 660-672, 2024.

[50] H. Al-Hamadi and S. Soliman, "Fuzzy short-term electric load forecasting using Kalman filter," *IEE Proceedings Generation, Transmission and Distribution,* vol. 153, no. 2, pp. 217-227, 2006.

[51] X. Chen, Y. Wang, and J. Tuo, "Short-term power load forecasting of GWO-KELM based on Kalman filter," *IFAC-PapersOnLine,* vol. 53, no. 2, pp. 12086-12090, 2020.

[52] T. Zheng, A. A. Girgis, and E. B. Makram, "A hybrid wavelet-Kalman filter method for load forecasting," *Electric power systems research,* vol. 54, no. 1, pp. 11-17, 2000.

[53] C.-N. Ko and C.-M. Lee, "Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter," *Energy,* vol. 49, pp. 413-422, 2013.

[54] M. S. ElMenshawy and A. M. Massoud, "Short-term load forecasting in active distribution networks using forgetting factor adaptive extended kalman filter," *IEEE Access,* vol. 11, pp. 103916-103924, 2023.

[55] U. D. Caprio, R. Genesio, S. Pozzi, and A. Vicino, "Short term load forecasting in electric power systems: a comparison of ARMA models and extended wiener filtering," *Journal of Forecasting,* vol. 2, no. 1, pp. 59-76, 1983.

[56] H. Sun, G. Feng, D. Nikovski, and J. Zhang, "Dynamic state estimation based

on unscented Kalman filter and very short-term load and distributed generation forecasting," in *Proc. 2016 IEEE International Conference on Power System Technology*, 2016, pp. 1-6.

[57] S. K. Dash, R. Bisoi, and P. Dash, "A hybrid functional link dynamic neural network and evolutionary unscented Kalman filter for short-term electricity price forecasting," *Neural Computing and Applications,* vol. 27, no. 7, pp. 2123-2140, 2016.

[58] T. Launay, A. Philippe, and S. Lamarche, "On particle filters applied to electricity load forecasting," *Journal de la Société Française de Statistique,* vol. 154, no. 2, pp. 1-36, 2013.

[59] F. He, J. Zhou, Z.-k. Feng, G. Liu, and Y. Yang, "A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm," *Applied energy,* vol. 237, pp. 103-116, 2019.

[60] I. García, S. Huo, R. Prado, and L. Bravo, "Dynamic Bayesian temporal modeling and forecasting of short-term wind measurements," *Renewable energy,* vol. 161, pp. 55-64, 2020.

[61] Q.-A. Wang *et al.*, "Bayesian dynamic linear model framework for structural health monitoring data forecasting and missing data imputation during typhoon events," *Structural Health Monitoring,* vol. 21, no. 6, pp. 2933-2950, 2022.

[62] L. Breiman, "Random forests," *Machine learning,* vol. 45, no. 1, pp. 5-32, 2001.

[63] Y.-Y. Song and Y. Lu, "Decision tree methods: applications for classification and prediction," *Shanghai Archives of Psychiatry,* 2015.

[64] A. Lahouar and J. B. H. Slama, "Day-ahead load forecast using random forest and expert input selection," *Energy Conversion and Management,* vol. 103, pp. 1040-1051, 2015.

[65] S. S. Subbiah and J. Chinnappan, "Short-term load forecasting using random forest with entropy-based feature selection," in *Artificial Intelligence and Technologies: Select Proceedings of ICRTAC-AIT 2020*: Springer, 2021, pp. 73-

80.

[66] G. Dudek, "Short-term load forecasting using random forests," in *Proc. 7th IEEE International Conference Intelligent Systems*, 2015, pp. 821-828.

[67] B. Magalhães, P. Bento, J. Pombo, M. d. R. Calado, and S. Mariano, "Short-term load forecasting based on optimized random forest and optimal feature selection," *Energies,* vol. 17, no. 8, p. 1926, 2024.

[68] F. Liu, T. Dong, T. Hou, and Y. Liu, "A hybrid short-term load forecasting model based on improved fuzzy c-means clustering, random forest and deep neural networks," *IEEE Access,* vol. 9, pp. 59754-59765, 2021.

[69] G.-F. Fan, L.-L. Peng, and W.-C. Hong, "Short-term load forecasting based on empirical wavelet transform and random forest," *Electrical Engineering,* vol. 104, no. 6, pp. 4433-4449, 2022.

[70] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Applied Energy,* vol. 132, pp. 602-609, 2014.

[71] Y. Zheng, L. Zhu, and X. Zou, "Short-term load forecasting based on gaussian wavelet SVM," *Energy Procedia,* vol. 12, pp. 387-393, 2011.

[72] J. Gao *et al.*, "Multiple kernel support vector machine short-term load forecasting based on multi-source heterogeneous integration of load factors," in *Proc. 2016 International Conference on Advanced Electronic Science and Technology*, 2016: Atlantis Press, pp. 10-22.

[73] Y. He, R. Liu, H. Li, S. Wang, and X. Lu, "Short-term power load probability density forecasting method using kernel-based support vector quantile regression and Copula theory," *Applied Energy,* vol. 185, pp. 254-266, 2017.

[74] S. Karthika, V. Margaret, and K. Balaraman, "Hybrid short term load forecasting using ARIMA-SVM," in *Proc. 2017 Innovations in Power and Advanced Computing Technologies*, 2017, pp. 1-7.

[75] W. Zhao, H. Xu, P. Chen, J. Zhang, J. Li, and T. Cai, "Elastic momentum-enhanced adaptive hybrid method for short-term load forecasting," *Energies,*

vol. 18, no. 13, p. 3263, 2025.

[76] Y. Liang, D. Niu, M. Ye, and W.-C. Hong, "Short-term load forecasting based on wavelet transform and least squares support vector machine optimized by improved cuckoo search," *Energies,* vol. 9, no. 10, p. 827, 2016.

[77] Y. Dai and P. Zhao, "A hybrid load forecasting model based on support vector machine with intelligent methods for feature selection and parameter optimization," *Applied energy,* vol. 279, p. 115332, 2020.

[78] R. Seunghyoung, N. Jaekoo, and K. Hongseok, "Deep neural network based demand side short term load forecasting," in *2016 IEEE International Conference on Smart Grid Communications*, 2016, pp. 308-313.

[79] W. Chandramitasari, B. Kurniawan, and S. Fujimura, "Building deep neural network model for short term electricity consumption forecasting," in *Proc. 2018 International Symposium on Advanced Intelligent Informatics*, 2018, pp. 43-48.

[80] L. Li, K. Ota, and M. Dong, "Everything is image: CNN-based short-term electrical load forecasting for smart grid," in *Proc. 14th international symposium on pervasive systems, algorithms and networks & 2017 11th international conference on frontier of computer science and technology & 2017 third international symposium of creative computing*, 2017, pp. 344-351.

[81] A. M. Tudose, D. O. Sidea, I. I. Picioroaga, V. A. Boicea, and C. Bulac, "A CNN based model for short-term load forecasting: a real case study on the Romanian power system," in *Proc. 55th International Universities Power Engineering Conference*, 2020, pp. 1-6.

[82] C. Lang, F. Steinborn, O. Steffens, and E. W. Lang, "Applying a 1D-CNN network to electricity load forecasting," in *Proc. International Conference on Time Series and Forecasting*, 2019, pp. 205-218.

[83] C. Lang, F. Steinborn, O. Steffens, and E. W. Lang, "Electricity Load Forecasting--An Evaluation of Simple 1D-CNN Network Structures," *arXiv preprint arXiv:1911.11536,* 2019.

[84] A. Kaligambe and G. Fujita, "Short-term load forecasting for commercial buildings using 1D convolutional neural networks," in *Proc. 2020 IEEE PES/IAS PowerAfrica*, 2020, pp. 1-5.

[85] B. Zhang, J.-L. Wu, and P.-C. Chang, "A multiple time series-based recurrent neural network for short-term load forecasting," *Soft Computing,* vol. 22, no. 12, pp. 4099-4112, 2018.

[86] H. Eskandari, M. Imani, and M. P. Moghaddam, "Convolutional and recurrent neural network based model for short-term load forecasting," *Electric Power Systems Research,* vol. 195, p. 107173, 2021.

[87] S. Muzaffar and A. Afshari, "Short-term load forecasts using LSTM networks," *Energy Procedia,* vol. 158, pp. 2922-2927, 2019.

[88] J. Lin, J. Ma, J. Zhu, and Y. Cui, "Short-term load forecasting based on LSTM networks considering attention mechanism," *International Journal of Electrical Power & Energy Systems,* vol. 137, p. 107818, 2022.

[89] B.-S. Kwon, R.-J. Park, and K.-B. Song, "Short-term load forecasting based on deep neural networks using LSTM layer," *Journal of Electrical Engineering & Technology,* vol. 15, no. 4, pp. 1501-1509, 2020.

[90] Z. Sheng, Z. An, H. Wang, G. Chen, and K. Tian, "Residual LSTM based short-term load forecasting," *Applied Soft Computing,* vol. 144, p. 110461, 2023.

[91] K. Ijaz, Z. Hussain, J. Ahmad, S. F. Ali, M. Adnan, and I. Khosa, "A novel temporal feature selection based LSTM model for electrical short-term load forecasting," *IEEE Access,* vol. 10, pp. 82596-82613, 2022.

[92] G. Xiuyun, W. Ying, G. Yang, S. Chengzhi, X. Wen, and Y. Yimiao, "Short-term load forecasting model of GRU network based on deep learning framework," in *Proc. 2nd IEEE Conference on Energy Internet and Energy System Integration*, 2018, pp. 1-4.

[93] J. Zheng, X. Chen, K. Yu, L. Gan, Y. Wang, and K. Wang, "Short-term power load forecasting of residential community based on GRU neural network," in *Proc. 2018 International Conference on Power System Technology*, 2018, pp.

4862-4868.

[94] S. Kumar, L. Hussain, S. Banarjee, and M. Reza, "Energy load forecasting using deep learning approach-LSTM and GRU in spark cluster," in *Proc. 5 international conference on emerging applications of information technology*, 2018: IEEE, pp. 1-4.

[95] S. H. Rafi, S. R. Deeba, and E. Hossain, "A short-term load forecasting method using integrated CNN and LSTM network," *IEEE Access,* vol. 9, pp. 32436-32448, 2021.

[96] Q. Fang, Y. Zhong, C. Xie, H. Zhang, and S. Li, "Research on pca-lstm-based short-term load forecasting method," in *Proc. IOP Conference Series: Earth and Environmental Science*, 2020, vol. 495, no. 1, p. 012015.

[97] Y. Xuan *et al.*, "Multi-model fusion short-term load forecasting based on random forest feature selection and hybrid neural network," *IEEE Access,* vol. 9, pp. 69002-69009, 2021.

[98] Finardi P, Campiotti I, Plensack G, de Souza RD, Nogueira R, and P. G. "Electricity theft detection with self-attention 2020." (accessed 08/11, 2025).

[99] S. Mitra, B. Chakraborty, and P. Mitra, "Smart meter data analytics applications for secure, reliable and robust grid system: Survey and future directions," *Energy,* vol. 289, p. 129920, 2024.

[100] H. AI. "Smart meter data analysis." (accessed 08/12, 2025).

[101] D. N. Karger, D. R. Schmatz, G. Dettling, and N. E. Zimmermann, "High-resolution monthly precipitation and temperature time series from 2006 to 2100," *Scientific Data,* vol. 7, no. 1, p. 248, 2020.

[102] S. Patidar, D. P. Jenkins, A. Peacock, and A. Lotfipoor, "Missing data imputation for community energy demand modelling," in *Proc. 3rd IBPSA-Scotland Conference*, 2022.

[103] R. Xiang, H. Yang, Z. Yan, A. M. Mohamed Taha, X. Xu, and T. Wu, "Super-resolution reconstruction of GOSAT CO2 products using bicubic interpolation," *Geocarto International,* vol. 37, no. 27, pp. 15187-15211, 2022.

[104] A. Allik and A. Annuk, "Interpolation of intra-hourly electricity consumption and production data," in *Proc. 2017 IEEE 6th international conference on renewable energy research and applications*, 2017: IEEE, pp. 131-136.

[105] A. Toktarova, L. Gruber, M. Hlusiak, D. Bogdanov, and C. Breyer, "Long term load projection in high resolution for all countries globally," *International Journal of Electrical Power & Energy Systems,* vol. 111, pp. 160-181, 2019.

[106] Y. Cui, R. Yan, R. Sharma, T. Saha, and N. Horrocks, "Realizing multifractality of smart meter data for household characteristic prediction," *International Journal of Electrical Power & Energy Systems,* vol. 139, p. 108003, 2022.

[107] X. Zhu and B. Mather, "DWT-based aggregated load modeling and evaluation for quasi-static time-series simulation on distribution feeders preprint," in *Proc. IEEE General Meeting Power& Energy Society*, 2018.

[108] M. Jiménez-Aparicio, M. J. Reno, and J. W. Pierre, "The high-resolution wavelet transform: A generalization of the discrete wavelet transforms," in *Proc. IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, 2022, pp. 0395-0401.

[109] Z. Li, B. Chang, S. Wang, A. Liu, F. Zeng, and G. Luo, "Dynamic compressive wide-band spectrum sensing based on channel energy reconstruction in cognitive internet of things," *IEEE Transactions on Industrial Informatics,* vol. 14, no. 6, pp. 2598-2607, 2018.

[110] C. Liu, A. Akintayo, Z. Jiang, G. P. Henze, and S. Sarkar, "Multivariate exploration of non-intrusive load monitoring via spatiotemporal pattern network," *Applied Energy,* vol. 211, pp. 1106-1122, 2018.

[111] *Pecan Street Dataport*. [Online]. Available: https://www.pecanstreet.org/dataport/

[112] I. de-Paz-Centeno, M. T. García-Ordás, O. García-Olalla, J. Arenas, and H. Alaiz-Moretón, "M-SRPCNN: A fully convolutional neural network approach for handling super resolution reconstruction on monthly energy consumption environments," *Energies,* vol. 14, no. 16, p. 4765, 2021.

[113] I. De-Paz-Centeno, M. T. García-Ordás, Ó. García-Olalla, and H. Alaiz-Moretón, "Using super resolution perception for anomaly detection in energy load profiles," in *Proc. 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2022, pp. 1-6.

[114] G. Liu *et al.*, "Super-resolution perception for wind power forecasting by enhancing historical data," *Frontiers in Energy Research,* vol. 10, p. 959333, 2022.

[115] T. Tong, L. Gen, L. Xiejie, and G. Qinquan, "A hybrid short-term load forecasting model based on improved fuzzy c-means clustering, random forest and deep neural networks," in *Proc. 2017 IEEE International Conference on Computer Vision*, 2017, pp. 4809-4817.

[116] Y. Jia, Z. Liang, X. Huo, W. Chen, Y. Chai, and R. Yu, "The impact of load-PV profile resolution on distribution system risk assessment," *Energy Reports,* vol. 9, pp. 2653-2664, 2023.

[117] G. Liang *et al.*, "Super resolution perception for improving data completeness in smart grid state estimation," *Engineering,* vol. 6, pp. 789-800, 2020.

[118] G. Liu, J. Gu, J. Zhao, F. Wen, and G. Liang, "Super resolution perception for smart meter data," *Information Sciences,* vol. 526, pp. 263-273, 2020.

[119] J. Ruan *et al.*, "Super-resolution perception assisted spatiotemporal graph deep learning against false data injection attacks in smart grid," *IEEE Transactions on Smart Grid,* vol. 14, no. 5, pp. 4035-4046, 2023.

[120] J. Huang, Q. Huang, G. Mou, and C. Wu, "DPWGAN: High-quality load profiles synthesis with differential privacy guarantees," *IEEE Transactions on Smart Grid,* vol. 14, pp. 3283 - 3295, 2022.

[121] F. Li, D. Lin, and T. Yu, "Improved generative adversarial network-based super resolution reconstruction for low-frequency measurement of smart grid," *IEEE Access,* vol. 8, pp. 85257-85270, 2020.

[122] L. Song, Y. Li, and N. Lu, "ProfileSR-GAN: A gan based super-resolution method for generating high-resolution load profiles," *IEEE Transactions on*

Smart Grid, vol. 13, pp. 3278-3289, 2022.

[123] R. Tang, J. Dore, J. Ma, and P. H. Leong, "Interpolating high granularity solar generation and load consumption data using super resolution generative adversarial network," *Applied Energy,* vol. 299, p. 117297, 2021.

[124] C. Zhang, Z. Shao, C. Jiang, and F. Chen, "A PV generation data reconstruction method based on improved super-resolution generative adversarial network," *International Journal of Electrical Power & Energy Systems,* vol. 132, p. 107129, 2021.

[125] Y. Zhao *et al.*, "Super-resolution reconstruction method for power system data considering weather impacts based on self-attention TimeGAN," in *Proc. IEEE/IAS 61st Industrial and Commercial Power Systems Technical Conference*, 2025, pp. 1-7.

[126] G. Gross and F. D. Galiana, "Short-term load forecasting," *Proceedings of the IEEE,* vol. 75, no. 12, pp. 1558-1573, 1987.

[127] N. Liu, J. Chen, L. Zhu, J. Zhang, and Y. He, "A key management scheme for secure communications of advanced metering infrastructure in smart grid," *IEEE Transactions on Industrial Electronics,* vol. 60, no. 10, pp. 4746-4756, 2013.

[128] M. T. Hagan and S. M. Behr, "The time series approach to short term load forecasting," *IEEE Transactions on Power Systems,* vol. 2, no. 3, pp. 785-791, 1987.

[129] F. Zhao and H. Su, "Short-term load forecasting using kalman filter and elman neural network," in *Proc. IEEE Conference on Industrial Electronics and Applications*, 2007, pp. 1043-1047.

[130] H. M. Al-Hamadi and S. A. Soliman, "Short-term electric load forecasting based on Kalman filtering algorithm with moving window weather and load model," *Electric Power Systems Research,* vol. 68, no. 1, pp. 47-59, 2004.

[131] G. Juberias, R. Yunta, J. G. Moreno, and C. Mendivil, "A new ARIMA model for hourly load forecasting," in *Proc. IEEE Transmission and Distribution*

Conference, 1999, vol. 1, pp. 314-319.

[132] W. R. Christiaanse, "Short-term load forecasting using general exponential smoothing," *IEEE Transactions on Power Apparatus and Systems,* vol. 90, no. 2, pp. 900-911, 1971.

[133] V. Mayrink and H. S. Hippert, "A hybrid method using exponential smoothing and gradient boosting for electrical short-term load forecasting," in *Proc. IEEE Latin American Conference on Computational Intelligence* 2016, pp. 1-6.

[134] J. Che and J. Wang, "Short-term load forecasting using a kernel-based support vector regression combination model," *Applied Energy,* vol. 132, no. 1, pp. 602-609, 2014.

[135] Z. Mingguang, "Short-term load forecasting based on support vector machines regression," in *Proc. International Conference on Machine Learning and Cybernetics*, 2005, vol. 7, pp. 4310-4314.

[136] G. Dudek, "Short-term load forecasting using random forests," in *Proc. Intelligent Systems*, 2015, pp. 821-828.

[137] A. Lahouar and J. Ben Hadj Slama, "Day-ahead load forecast using random forest and expert input selection," *Energy Conversion and Management,* vol. 103, no. 1, pp. 1040-1051, 2015.

[138] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems,* vol. 33, no. 1, pp. 1087-1088, 2018.

[139] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid,* vol. 10, no. 1, pp. 841-851, 2019.

[140] H. S. Hippert, D. W. Bunn, and R. C. Souza, "Large neural networks for electricity load forecasting: Are they overfitted?," *International Journal of Forecasting,* vol. 21, no. 3, pp. 425-434, 2005.

[141] Y. He, F. Luo, and G. Ranzi, "Transferrable model-agnostic meta-learning for short-term household load forecasting with limited training data," *IEEE*

Transactions on Power Systems, vol. 37, no. 4, pp. 3177 - 3180, 2022.

[142] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *Proc. IEEE International Conference on Communications*, 2020, pp. 1-6.

[143] Y. He, F. Luo, G. Ranzi, and W. Kong, "Short-term residential load forecasting based on federated learning and load clustering," in *Proc. IEEE International Conference on Smart Grid Communications*, 2021, pp. 77-82.

[144] N. Gholizadeh and P. Musilek, "Federated learning with hyperparameter-based clustering for electrical load forecasting," *Internet of Things,* vol. 17, no. 1, pp. 1-10, 2022.

[145] Y. L. Tun, K. Thar, C. M. Thwal, and C. S. Hong, "Federated learning based energy demand prediction with clustered aggregation," in *Proc. IEEE International Conference on Big Data and Smart Computing*, 2021, pp. 164-167.

[146] M. N. Fekri, K. Grolinger, and S. Mir, "Distributed load forecasting using smart meter data: federated learning with recurrent neural networks," *International Journal of Electrical Power & Energy Systems,* vol. 137, no. 1, pp. 1-12, 2022.

[147] M. Savi and F. Olivadese, "Short-term energy consumption forecasting at the edge: a federated learning approach," *IEEE Access,* vol. 9, pp. 95949-95969, 2021.

[148] M. Jia, Y. Wang, C. Shen, and G. Hug, "Privacy-preserving distributed clustering for electrical load profiling," *IEEE Transactions on Smart Grid,* vol. 12, no. 2, pp. 1429-1444, 2020.

[149] Y. Wang, M. Jia, N. Gao, L. Von Krannichfeldt, M. Sun, and G. Hug, "Federated clustering for electricity consumption pattern extraction," *IEEE Transactions on Smart Grid,* vol. 13, no. 3, pp. 2425-2439, 2022.

[150] A. Australian Govern., Canberra, ACT. *Smart Grid, Smart City*. [Online]. Available: https://data.gov.au/dataset.

[151] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[152] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. International Conference on Computational Statistics*, 2010, pp. 177-186.

[153] G. Chicco, "Overview and performance assessment of the clustering methods for electrical load pattern grouping," *Energy,* vol. 42, no. 1, pp. 68-80, 2012.

[154] A. S. Musleh, G. Chen, and Z. Y. Dong, "A survey on the detection algorithms for false data injection attacks in smart grids," *IEEE Transactions on Smart Grid,* vol. 11, no. 3, pp. 2218-2234, 2020.

[155] A. Capozzoli, M. S. Piscitelli, S. Brandi, D. Grassi, and G. Chicco, "Automated load pattern learning and anomaly detection for enhancing energy management in smart buildings," *Energy,* vol. 157, pp. 336-352, 2018.

[156] R. Yao and K. Steemers, "A method of formulating energy load profile for domestic buildings in the UK," *Energy and Buildings,* vol. 37, no. 6, pp. 663-671, 2005.

[157] N. S.-N. Lam, "Spatial interpolation methods: a review," *The American Cartographer,* vol. 10, no. 2, pp. 129-150, 1983.

[158] Z. Wang, Y. Chen, S. Huang, X. Zhang, and X. Liu, "Temporal graph super resolution on power distribution network measurements," *IEEE Access,* vol. 9, pp. 70628-70638, 2021.

[159] W. Liu, C. Ren, and Y. Xu, "PV generation forecasting with missing input data: A super-resolution perception approach," *IEEE Transactions on Sustainable Energy,* vol. 12, no. 2, pp. 1493-1496, 2020.

[160] C. Ren, Y. Xu, J. Zhao, R. Zhang, and T. Wan, "A super-resolution perception-based incremental learning approach for power system voltage stability assessment with incomplete PMU measurements," *CSEE Journal of Power and Energy Systems,* vol. 8, no. 1, pp. 76-85, 2021.

[161] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Conference on Computer Vision and Pattern Recognition*, 2014.

[162] C. K. Koç, "Analysis of sliding window techniques for exponentiation,"

Computers & Mathematics with Applications, vol. 30, no. 10, pp. 17-24, 1995.

[163] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 977-984.

[164] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift.," *arXiv preprint arXiv:1502.03167,* 2015.

[165] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874-1883.

[166] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. the European Conference on Computer Vision (ECCV)*, 2018, pp. 286-301.

[167] Y. Zhang, L. Dong, H. Yang, L. Qing, X. He, and H. Chen, "Weakly-supervised contrastive learning-based implicit degradation modeling for blind image super-resolution," *Knowledge-Based Systems,* vol. 249, p. 108984, 2022.

[168] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748,* 2018.

[169] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. The IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770-778.

[170] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, no. 11, pp. 1532-4435, 2008.

[171] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[172] "Ptflops." Available: https://pypi.org/project/ptflops (accessed Mar. 7, 2025).

[173] C. Zhang, J. Wu, Y. Zhou, M. Cheng, and C. Long, "Peer-to-Peer energy trading in a Microgrid," *Applied Energy,* vol. 220, pp. 1-12, 2018.

[174] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual

networks for single image super-resolution," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 136-144.

[175] M. Mohandes, "Support vector machines for short-term electrical load forecasting," *International Journal of Energy Research,* vol. 26, pp. 335-345, 2002.

[176] Y. Cheng, P. P. K. Chan, and Z. Qiu, "Random forest based ensemble system for short term load forecasting," in *2012 International Conference on Machine Learning and Cybernetics*, 2012, vol. 1, pp. 52-56.

[177] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Transactions on Smart Grid,* vol. 10, pp. 841-851, 2019.

[178] E. Lee and W. Rhee, "Individualized short-term electric load forecasting with deep neural network based transfer learning and meta learning," *IEEE Access,* vol. 9, pp. 15413-15425, 2021.

[179] A. Taïk and S. Cherkaoui, "Electrical load forecasting using edge computing and federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1-6.

[180] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.