Deep Learning-based Load Forecasting and Monitoring in the Smart Grid

by

Lingxiao Wang

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 11, 2021

Keywords: Load forecasting, Deep learning, Long Short-Term Memory (LSTM), Smart Grid, Ensemble learning, Meta-learning, Transfer learning, Pre-trained model, Non-intrusive load monitoring, Transformer.

Approved by

Shiwen Mao, Chair, Professor of Electrical and Computer Engineering
Xiaowen Gong, Assistant Professor of Electrical and Computer Engineering
Mark Nelms, Professor of Electrical and Computer Engineering
Thaddeus Roppel, Associate Professor of Electrical and Computer Engineering
Tao Shu, Associate Professor of Computer Science and Software Engineering

Abstract

With the rapid advances in sensing and acquisition, transmission, storage, computing, and analytics, the era of big data has come. Many advanced data analytic techniques, especially machine learning and deep learning techniques, have been proposed and found wide applications in our society.

In the power industry, data analytics play an essential role in daily power system operation and planning. One major challenge for energy management in the emerging smart grid is the uncertainty in both power supply (e.g., renewable energy generation) and demand (e.g., load demand from the service area). There is a compelling need to accurately predict generation and load for efficient power management. Such predictions will help to make intelligent decisions for improving power quality, saving energy, better utilizing renewable energy sources, and reducing cost.

This thesis develops data-driven solutions by using the latest deep learning and machine learning technology, including ensemble learning, meta-learning, and transfer learning, for energy management system issues, such as short-term load forecasting and non-intrusive load monitoring problems. Real-world datasets are tested on proposed models compared with state-of-the-art schemes, which demonstrates the superior performance of the proposed model.

Acknowledgments

I treasure the process of fulfilling my Ph.D. degree at Auburn University, which I will never forget. I owe thanks to all who made this journey possible. At first, I would like to thank my advisors, Dr. Shiwen Mao and Dr. Bogdan M.Wilamowski, who patiently guide, continuously support, and unlimitedly encourage an ignorant student to pursue knowledge. Both of you gave me the freedom to follow my interests. You taught me how to be creative and optimistic during my Ph.D. study, which helped me a lot as this journey is full of challenges.

I also owe special thanks to Prof. Xiaowen Gong, Prof. Thaddeus Roppel, and Prof. Mark Nelms, who serve as my dissertation committee. They provided invaluable support and guidance on my dissertation. Besides, I would like to thank Dr. Tao Shu for the comments on my dissertation.

Mom and Dad, your love made me who I am today. Thank you for offering an opportunity to study abroad. I want to thank my lab mates, Bo Wu, Xi Meng, Xiangyu Wang, Chao Yang, and Ticao Zhang. It is my honor to work with you all. Last but not least, I would like to thank my friends, Xinyu Zhu, Xing Wang, Yijun Wang, Zhuonan Wang, Zhan Su, Yingzhou Yun, Cunxia Han, Yana Lou, Yuning Zhao, Hanlu Yu, Han Zhang, Fu Zhao, Anni Zhang, Wenshan Liu, Tian Liu, and Yecheng Xu.

This work is supported in part by the US National Science Foundation (NSF) under Grant CNS-2107190, DMS-1736470, and by the Wireless Engineering Research and Education Center (WEREC) at Auburn University, Auburn, AL, USA.

Table of Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Rapid progress in urbanization brings about significant changes in people's lifestyles. In light of this trend, many challenging problems - such as environmental pollution, traffic problems, high energy consumption, and so on - are raised. In order to address these issues, the concept of urban computing is introduced, which involves collecting, integrating, and analyzing the data generated by devices in an urban area to improve people's life quality [6, 7]. With the fast development of artificial intelligence, machine learning, in particular, deep learning, techniques show high potential for addressing many urban computing problems. This is mainly due to the breakthroughs in computing and the rapid advances in sensing and data acquisition, transmission, and storage [8]. Researchers now have the capability of handling large-scale data and utilizing it more wisely.

Today's sustainable urban power systems, i.e., the smart grid, are characterized by high energy efficiency, demand-side management, renewable energy sources, and a two-way flow of information and electricity, as enabled by the integration of communications, control, and signal processing [9].

The evolution of generation, transmission, operation, and consumption significantly affects smart grid development, impacting its planning and operation, which brings new perspectives to energy management and demand response in the smart grid [10–13]. New and clean devices, as well as modern technologies in data analytics, communication systems, control, and information theory, enable an advanced power system with higher energy efficiency and power delivery stability [14, 15].

1.1 Deep Learning and Data Analytics for Smart Meters

Smart metering or information metering plays an essential role in information acquisition from end power users, which automatically collects data from all the metering equipment in power grids [14, 16]. Smart meters are electric meters that record the power consumption information of end-users and support the two-way information flows with the control center. Smart meters have been deployed rapidly during the past decade. According to [17], about 100 million smart meters will have been installed in the US by the end of 2021.

On the one hand, for retailers, communication networks collect data and information from the power grid components, which can be analyzed and used to control the power system for real-time pricing, demand response, and protection [18]. On the other hand, for consumers, networks construct communication paths that integrate smart meters, home appliances, and renewable energy sources for Home Energy Management Systems (HEMS) [19, 20].

With the continuous development of computing power, deep learning stands out among artificial intelligence (AI) and has flourished. It (DL) is based on the core of Artificial Neural Networks (ANNs) or Multilayer Perceptrons (MLPs), which is inspired by the biological neurons. Deep Neural Networks (DNNs) acquire increasing learning capacity by adding more layers or units within a layer. Different types of DNNs address different tasks. For example, Convolutional Neural Networks (CNNs) extract features and patterns within an input, which is suitable for headling computer vision problems. Recurrent Neural Networks (RNNs) are designed to process sequential data, introducing state storing past information. It leverages text, audio, and times series data.

Deep learning has already been primarily applied in energy management with the help of smart meter data. The main application can be summarized into three aspects: load analysis, load forecasting, and load management. This dissertation focuses on load forecasting at the system, zone, client, and appliance levels. Specifically, we tried to solve short-term load forecasting and Non-intrusive load monitoring (NILM) problems by deep learning.

1.2 Outline of Chapters

In Chapter 2, an ensemble learning approach is proposed for load forecasting in urban power systems. The proposed framework consists of two levels of learners that integrate clustering, Long Short-Term Memory (LSTM) and a Fully Connected Cascade (FCC) neural network. A clustering algorithm first partitions historical load data to train multiple LSTM models in the level-one learner, and then the FCC model in the second level is used to fuse the multiple level-one models. A modified Levenberg-Marquardt (LM) algorithm trains the FCC model for fast and stable convergence. The proposed framework is tested with two public datasets for short-term and mid-term forecasting at the system, zone, and client levels. In Chapter 3, we propose a pre-training approach to address the generalization of DL models for NILM. We develop a meta-learning-based approach and an ensemble learning-based approach, which pre-trains a base model and then fine-tunes it with few-short learning when applied to an unknown dataset. The models are validated with two real-world datasets and achieved a superior transferability performance compared with traditional DL and transfer learning methods. In Chapter 4, we propose a Middle Window Transformer, termed Midformer, for NILM tasks. Existing models are limited by high computational complexity, dependency on data, and poor transferability. In Midformer, we first exploit patch-wise embedding to shorten the input length, and then reduce the size of queries in the attention layer, by only using global attention on a few selected input locations at the center of the window to capture global contexts. The cyclically shifted window technique is used to preserve connection across patches. We also follow the pre-training and fine-tuning paradigm to relieve the dependency on data, reduce the computation in modeling training, and enhance transferability of the model to unknown tasks and domains. Our experimental study using two real-world datasets demonstrates the superior performance and great transferability of Midformer over three baseline models. Chapter 5 concludes the research. Future works are also discussed.

Chapter 2

Ensemble Learning for Load Forecasting

2.1 Load forecasting

For load forecasting, many methods have been proposed for it. Machine learning and statistical methods are the two main approaches that are widely applied. For example, in [21], the authors propose an ensemble approach based on extreme learning machine for short-term load forecasting. Radial Basis Function (RBF) neural networks trained with a second-order algorithm are utilized in [22] for short-term load forecasting. These two schemes both have a shallow structure in their neural network design. Deep learning has become a hot technique due to their recent demonstrated success in computer vision and natural language processing (NLP). Among various deep learning models, recurrent neural networks, e.g., Long Short Term Memory (LSTM), has been proposed for handling residential data in [23,24]. It is shown in [23] that an LSTM-based Sequence to Sequence (S2S) architecture can handle both one-minute and one-hour resolution data for one residential customer. In [24], the authors focus on short-term forecasting individual customer's consumption of power using LSTM. Effectiveness of accurate short-term load forecasting has been demonstrated in [25] by using a Deep Residual Network (res-net). In addition, Quantile Regression is a popular statistic technique for load forecasting. In [26], the authors exploit the quantile regression model to enhance forecasting performance. In [27], the authors improve the traditional quantile regression neural network and demonstrate its reliability in probabilistic load forecasting.

In this chapter, an ensemble learning approach is proposed to tackle the load forecasting problem. Our proposed framework consists of two levels of learners. The first-level learner utilizes the

LSTM model to obtain the first-level predictions, while a fully connected cascade (FCC) neural networks are incorporated in the second-level learner for the purpose of model fusion. Our proposed framework has three notable features. First, point load forecasting is a regression problem, to which unsupervised learning techniques can be easily applied. The proposed framework integrates unsupervised learning with a supervised learning model for accurate load prediction, which is a novel approach comparing to existing load forecasting models. Specifically, clustering algorithms are incorporated in our framework, to partition data into individual clusters according to their similarity. Each data cluster is then used to generate an LSTM base model to obtain the first-level prediction. Then the first-level prediction results are fused by the second-level FCC neural network as supervised learning to enhance the accuracy of load forecasting.

Second, for various learning problems, a deep neural network may not always be the chosen one; it is critical to choose the right neural network structure properly. In this work, we select a deep (LSTM) and a shallow (FCC) structure in the two different levels of learning, respectively. It is well-known that the deeper the neural network, the more likely overfitting will occur. Thus, it is highly desirable to have a learner that can provide a sufficient learning ability, while using as few layers as possible. In the proposed framework, the first-level learner captures most of the nonlinear relationship between input and output data, while the second-level learner discerns the linear connection between them. This is the criterion that guides our choice of proper neural architecture in the proposed framework. Third, ensemble learning is used in the proposed framework. The boosted fusion model (ensemble) in the second level enhances the accuracy of load prediction [28].

The contributions can be briefly summarized as follows. First, an ensemble learning approach is proposed to integrate state-of-the-art machine learning algorithms, i.e., clustering, LSTM, and FCC, for accurate load forecasting. We also study four different, representative clustering algorithms applied in the first level of learning and found the integration of HDBSCAN and LSTM achieve the best performance. Second, we propose to use an FCC neural network for model fusion in the second-level learner and a fast converging and stable modified Levenberg-Marquardt (LM) optimization algorithm for training the second-level learner. The FCC network captures the

relationship among individual models and thus improve the prediction accuracy. Third, we validate our proposed framework with two public datasets and compare its performance with several state-of-the-art schemes, where superior performance is demonstrated for the proposed framework. Fourth, the proposed framework can effectively deal with both short-term (e.g., hour-ahead) and mid-term (e.g., week-ahead) load forecasting, for not only system-level but also zone-level and client/residential-level forecasting.

2.2 The Proposed Framework

In this section, we first formulate the power load forecasting problem. We then discuss the details of our proposed framework in the remainder of the section, including the design of the two levels of learners.

2.2.1 Problem Statement

In this chapter, we focus on the load forecasting problem. Consider a time series signal $\boldsymbol{Y}_T = \{\boldsymbol{f}_1, \boldsymbol{f}_2, ..., \boldsymbol{f}_{m-1}, \boldsymbol{\ell}\}$, where $\boldsymbol{Y}_T \in \mathbb{R}^{m \times T}$. $\boldsymbol{Y}_T$ consists of two components, i.e., the feature part and load part. In the feature part, $\boldsymbol{f}_i = \{f_{i1}, f_{i2}, ..., f_{iT}\}$, which is the historical data of the ith feature that affects load. For example, temperature is one of the most important features that affect the power load. If features are not provided in the dataset, this part would set to null, and the forecasting will use historical load data only. The load part consists of $\boldsymbol{\ell} = \{\ell_1, \ell_2, ..., \ell_T\}$, i.e., the historical load data.

The goal is to forecast the load at a future time $T + \tau$ in a rolling predicting fashion, where τ is the amount of time ahead of the current time T. That is, we assume that only the information at and before T, i.e., $\boldsymbol{Y}_t$, for $t \leq T$, is available when predicting $\ell_{T+\tau}$. For example, to forecast the load value at time $T + 1$ (i.e., one time step ahead), $\boldsymbol{Y}_T$ is available and used. In order to ease training and reduce the training time, a window filter W is applied to $\boldsymbol{Y}_T$, which stores only the

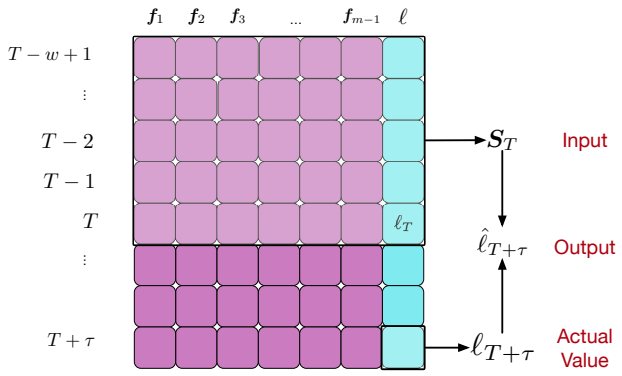


Figure 2.1: Input and output of the proposed neural network model.

data for w time steps, from the current time T back to time $T-w+1$. The input matrix $\boldsymbol{S}_T$ is thus defined as $\boldsymbol{S}_T = W(\boldsymbol{Y}_T)$, which is an $m \times w$ matrix.

Fig. 2.1 presents the mechanism of window filter and the formation of input and output data. The forecast value $\hat{\ell}_{T+\tau}$ is obtained by a fitting function as

$$\hat{\ell}_{T+\tau} = g(\boldsymbol{S}_T). \tag{2.1}$$

The goal of our proposed machine learning based predictive method is to learn the fitting function $g(\cdot)$ from the dataset $\boldsymbol{Y}_T$ that is available.

2.2.2 The Proposed Ensemble Learning Framework

To achieve high accuracy of power load forecasting, the concept of *stacking* is incorporated in our framework [29]. Stacking is a procedure of first training individual machine learning models and then integrating them [28]. There are two levels of learners in our proposed framework, where the first-level learner consists of multiple *individual learning models* and the second-level learner is used to combine the outputs from the individual learners in the first level for an integrated output. In order to meet the feature of stacking and testing, the data should first be divided into three parts. The first-level learners use the first part of data (denoted by $D1$). After the First-level learning models are built and trained, new data are generated from this level of learner, which is combined

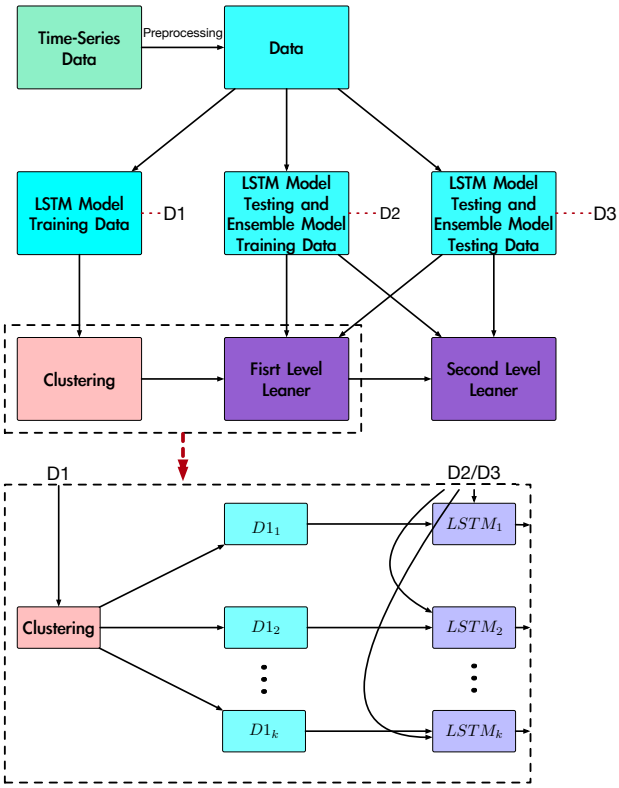


Figure 2.2: The proposed load prediction framework with two levels of learners.

with the second and third parts of data (denoted by $D2$ and $D3$, respectively). The combined two parts of data are used to train the second-level learner and test the framework.

In this chapter, we propose to use LSTM a recurrent neural network model, for the first-level learning and the FCC neural network for second-level learning. Fig. 2.2 illustrates the structure of the proposed framework. After preprocessing, the dataset is clustered into three parts, $D1$, $D2$, and $D3$ for training and testing purposes. The proposed predictor consists of a clustering algorithm, a set of LSTM models in the first-level learner, and an FCC model in the second-level learner. We discuss the design of these components in detail in the rest of this section.

2.2.3 First Level Learner

The first-level learner consists of a set of LSTM predictive models as well as a clustering algorithm, whose procedure is presented in Algorithm 1. The clustering algorithm partition the input data $D1$ into $D1_1, D1_2, ..., D1_k$, each being used to train an individual LSTM model.

Algorithm 1: Build the First Level LSTM Predictors

1 Partition the input data in dataset $D1$ into k clusters using a clustering algorithm ;
2 Divide dataset $D1$ into k individual datasets (i.e., including both input/output data) according to the clustering results: $\{\mathrm{D1}_1, \mathrm{D1}_2, ..., \mathrm{D1}_k\}$, where $\mathrm{D1}_i$ is the ith dataset produced by the ith cluster ;
3 Use each dataset $\mathrm{D1}_i$, $i = 1, 2, ..., k$ to train an individual LSTM model i ;

Clustering

Before data can be used by the LSTM models, we employ a clustering algorithm to partition the dataset based on the similarity among input data samples. Clustering is usually an unsupervised machine learning technique, referring to the process of grouping unlabeled data into clusters of similar features [30].

Note this is different from classification, which is based on given labeled data. It is well-known that the electricity demand is correlated with various obvious factors, such as temperature and calendar dates (e.g., weekday, holiday, month, season, etc.), while also being affected by uncertainties or latent factors as well.

We propose the use of unsupervised learning in our forecasting model with the following reasons. First of all, group input data of load forecasting into suitable sets and use different learning model for each set, are beneficial to better explore the correlation in the dataset [31]. Second, we assume that short term load variations are affected by the historical data of the time immediately before the current time. With unlabeled historical electric load data, clustering can group the data samples automatically and reasonably. Last but not least, partitioning the training dataset first and combining the learning results from the models later, resembles a kind of resampling process. This is similar to the process of cross-validation technique, which can mitigate the overfitting problem in machine learning.

Table 2.1: Clustering Algorithms used in This Chapter

Partitioning	
K-Means++ [33]	1.1 Choose seeds (i.e., the initial cluster centers) for K-means 1.2 Improve speed and accuracy of K-means
Hierarchical	
BIRCH [34]	2.1 Balanced Iterative Reducing and Clustering using Hierarchies 2.2 Based on the concept of Clustering Feature (CF) and CF tree 2.3 Does not need a predetermined number of clusters k 2.4 Can remove noise (outliers)
Density Based	
DBSCAN [35]	3.1 Density Based Spatial Clustering of Application with Noise 3.2 Uses parameters $(\epsilon, Minpts)$ to characterize the density of the data space
HDBSCAN [36, 37]	4.1 Hierarchical DBSCAN 4.2 Removes border points in DBSCAN 4.3 Superior to DBSCAN from a qualitative clustering perspective [38]

Clustering Algorithms

The collected power load time series data is usually susceptible to noise, shifting, and deformation [32]. It is important to choose an appropriate clustering method, from various existing techniques, to handle such data. In this chapter, we choose four representative algorithms from three categories of clustering methods, i.e., (i) partitioning methods, (ii) hierarchical methods, and (iii) density based methods. The chosen methods are K-means++ [33], BIRCH [34], DBSCAN [35], and HDBSCAN [36, 37], as summarized in Table 2.1. Note that for DBSCAN and HDBSCAN, some data samples are identified as outliers. Such group of outlier data is treated as one unique cluster in our proposed framework.

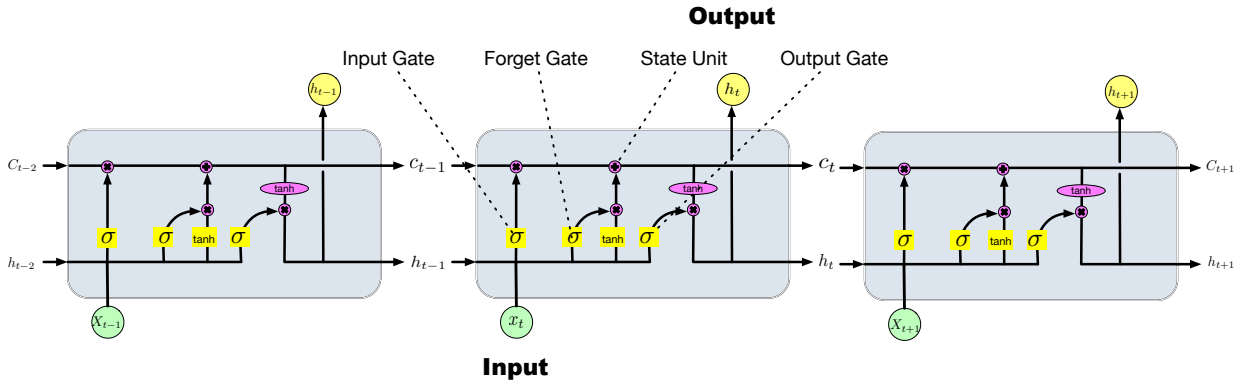


Figure 2.3: An unfolded view of the LSTM neural cell structure.

Long Short-Term Memory (LSTM)

Inspired by the novel idea of using three types of gates to regulate information flow and remembering information for over an arbitrary time interval [39], LSTM overcomes the limitation of long memory capability in recurrent neural networks. An unfolded illustration of the LSTM neural network is presented in Fig. 2.3. Input gate i_t, forget gate f_t, output gate o_t, and state unit c_t are the four key components in each LSTM cell (for time t). The state of LSTM cell at time t is calculated as

$$i_t = \sigma(\boldsymbol{W}^i h_{t-1} + \boldsymbol{U}^i x_t + \boldsymbol{b}^i) \tag{2.2}$$

$$f_t = \sigma(\boldsymbol{W}^f h_{t-1} + \boldsymbol{U}^f x_t + \boldsymbol{b}^f) \tag{2.3}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \sigma(\boldsymbol{W} h_{t-1} + \boldsymbol{U} x_t + \boldsymbol{b}) \tag{2.4}$$

$$o_t = \sigma(\boldsymbol{W}^o h_{t-1} + \boldsymbol{U}^o x_t + \boldsymbol{b}^o) \tag{2.5}$$

$$h_t = \tanh(c_{t-1}) \cdot o_t. \tag{2.6}$$

In the training phase, each LSTM model $LSTM_i$ will be trained with the corresponding data cluster $D1_i$, $i = 1, 2, ..., k$, as shown in Fig. 2.2.

Testing Process in the First Level Learner

During the training phase for the level two learner and the testing phase, new input data samples beyond $D1$ (i.e., in $D2$ and $D3$, respectively) arrives and are fed into the first-level learner. How to deal with them should be carefully designed. One way is to select the most similar cluster and use the corresponding trained LSTM model as in our prior work [31]. In this chapter, however, we propose to use *ensemble learning*, which is based on the assumption that power load prediction is driven by each of the homogeneous first level models. Thus the new data sample is fed into each first-level LSTM model, and an FCC neural network is used in the second level to fuse the outputs from the LSTM models to produce a single prediction.

2.2.4 Second Level Learner

Dataset $D2$ is used to train the second-level learner. Specifically, the data samples in $D2$ are first fed into each trained LSTM predictors in the first-level. Each LSTM predictor then generates a prediction value. These outputs are used as input to train the second-level learner.

The FCC neural network is incorporated for ensemble learning at level two. Fig. 2.4 shows an example of the FCC ensemble neural network. In this example, k base models are available and to be fused by five neurons. The first four neurons are activated by the $\tanh(\cdot)$ activation function, given by $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$. The last neuron is a linear summation. With the same number of neurons in level two, the FCC neural network architecture is superior to traditional neural network structures [40], as it provides more connections (and weights) than the traditional architecture, which make it deeper. The FCC neural network is similar to Deep Residual Networks [41] in some sense, which has an identity mapping for every input and latent variable to every neuron.

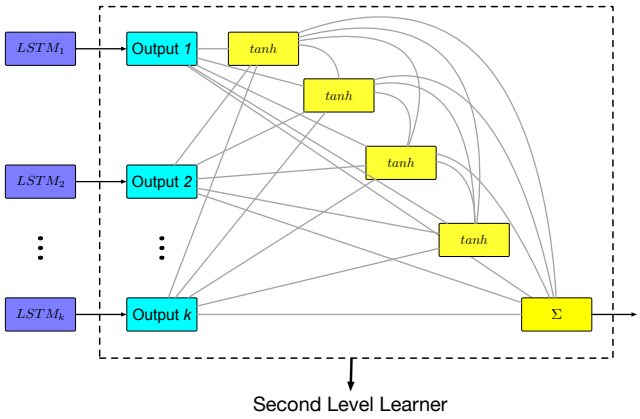


Figure 2.4: An example of FCC ensemble neural network used in the second-level learner.

2.3 Optimization and Training

2.3.1 Problem Formulation

We use the sum square error as the default lost function for the two levels of learners. The corresponding objective function of the LSTM model i at lever one is defined as

$$\mathcal{L}(L_1; i) = \underset{\boldsymbol{\omega}^i_{lstm}}{\text{minimize}} \sum_{T \in D1_i} \|\hat{\ell}^{L_1,i}_{T+\tau} - \ell_{T+\tau}\|^2 + \alpha \cdot \|\boldsymbol{\omega}^i_{lstm}\|, \tag{2.7}$$

where $\hat{\ell}^{L_1,i}_{T+\tau}$ is the predicted value of load by LSTM model i for time $T+\tau$, $\ell_{T+\tau}$ is the ground truth (i.e., label), and $\boldsymbol{\omega}^i_{lstm}$ are the wights of LSTM model i at the first level.

Supposing there are k trained LSTM models in the first-level learner, the load predicted by the level-two learner at time $T+\tau$ is given by

$$\hat{\ell}^{L_2}_{T+\tau} = f\left(\hat{\ell}^{L_1;1}_{T+\tau}, \hat{\ell}^{L_1;2}_{T+\tau}, ..., \hat{\ell}^{L_1;k}_{T+\tau}; \boldsymbol{\omega}_{fcc}\right), \tag{2.8}$$

where $f()$ is the output of the ensemble FCC neural network, $\hat{\ell}^{L_1;i}_{T+\tau}$ is the load forecast value predicted by LSTM model i, and $\boldsymbol{\omega}_{fcc}$ are the weights of the ensemble FCC neural network. The corresponding optimization objective over the validation and ensemble dataset D2 in level two is

given by

$$\mathcal{L}(L_2) = \underset{\boldsymbol{\omega}_{fcc}}{\text{minimize}} \sum_{T \in D2} \|\hat{\ell}^{L_2}_{T+\tau} - \ell_{T+\tau}\|^2 + \beta \cdot \|\boldsymbol{\omega}_{fcc}\|. \tag{2.9}$$

In both the first-level and second-level optimization objective functions, the L1 regulation is used to prevent overfitting in the neural network training process.

2.3.2 Gradient Descent Algorithms

First-order gradient descent algorithms, such as error backpropagation, Stochastic Gradient Decent (SGD), and its variants Adam, are quite successful in training deep neural networks. However, ill-conditioning and local-minima are common challenges for these algorithms. In [42], a second-order gradient descent algorithm is proved as an effective solution for optimizing problems with an objective function that exhibits pathological curvature. However, the second-order gradient descent algorithm also has its limitations. One challenge is that, for very deep neural networks, the second-order algorithm calculates the Hessian Matrix of the neural network, which takes a relatively longer period of time to train. The other issue is that, as the number of layers is increased, the large values of weights may get stuck in the saturated region, whose derivative of gradient tends to zero, and thus causing a vanishing gradient condition (known as the flat-spot problem) [43].

Given all the advantages and disadvantages of second-order gradient descent algorithms, we choose to apply the Adam algorithm [44], which is a first order gradient-based algorithm, to solve the regression task problem at level one, due to its deep structure. At level two, where FCC is a shallow neural network, we utilize the modified Levenberg-Marquardt (LM) Algorithm [45], which is a second-order optimization algorithm. The reason for a shallow architecture is applied at level two is that, we aim to provide a sufficient learning capacity for the training samples with the least number of neurons to overcome the overfitting problem.

Algorithm 2: The Modified Levenberg MarQuardt Method

1 Set $0 < m < \alpha_1$ and $0 < p_0 < p_1 < p_2 < 1$, where $\alpha_1 = 10^{-6}$, $m = 10^{-7}$, $p_0 = 10^{-4}$, $p_1 = 0.2$, $p_2 = 0.8$, and $e = 1$;

2 Calculate Jacobian Matrix $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})$ and approximate the Hessian matrix of the FCC neural network at the second level at iteration $e = 1$;

3 The normal LM step as $\boldsymbol{d}_e = \Delta\boldsymbol{\omega}_{fcc}^{e}$;

4 A line search for approximating the LM step $\Delta\boldsymbol{\omega}_{fcc}^{e'}$;

5 Combine Steps 2 and 3 as $\boldsymbol{s}_e = \Delta\boldsymbol{\omega}_{fcc}^{e} + \alpha_e\Delta\boldsymbol{\omega}_{fcc}^{e'}$;

6 If $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})^T\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e}) = 0$, then stop ;

7 Compute $r_e = R_a^e/R_p^e$, and set

$$\boldsymbol{\omega}_{fcc}^{e+1} = \begin{cases} \boldsymbol{\omega}_{fcc}^{e} + \boldsymbol{s}_e, & \text{if } r_e > p_0 \\ \boldsymbol{\omega}_{fcc}^{e}, & \text{otherwise ;} \end{cases} \tag{2.10}$$

8 Compute

$$\alpha_{e+1} = \begin{cases} 4\alpha_e, & \text{if } r_e < p_1 \\ \alpha_e, & \text{if } r_e \in [p_1, p_2] \\ \max(0.25\alpha_e, m), & \text{if } r_e > p_1 \text{ ;} \end{cases} \tag{2.11}$$

9 Set $e = e + 1$, and go to Step 2 ;

2.3.3 Modified Levenberg-Marquardt (LM) Algorithm

In this section, we introduce how to apply the modified LM in training the ensemble neural network at level two. The procedure is presented in Algorithm 2. The convergence of this method is proven in [45, 46].

The Jacobian Matrix $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})$ at iteration e is calculated by the derivative of (2.9), which is given by

$$\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e}) = \left[\frac{\partial\mathcal{L}(L_2)}{\partial\omega_1^e}, \frac{\partial\mathcal{L}(L_2)}{\partial\omega_2^e}, ..., \frac{\partial\mathcal{L}(L_2)}{\partial\omega_Z^e}\right], \tag{2.12}$$

where $\boldsymbol{\omega}_{fcc}^{e}$ is the weights of the FCC neural network at iteration e, which has Z weight values denoted by $\{\omega_1^e, \omega_2^e, ..., \omega_Z^e\}$. The Hessian matrix can be approximated by $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^{e})^T$. A

damping factor μ_e is updated iteratively as

$$\mu_e = \alpha_e \|\mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e)\|^\beta, \tag{2.13}$$

where $\beta \in (0, 2]$. At each iteration, the weights of the FCC neural network are updated as

$$\boldsymbol{\omega}_{fcc}^{e+1} = \boldsymbol{\omega}_{fcc}^e + \boldsymbol{s}_e, \tag{2.14}$$

or

$$\boldsymbol{\omega}_{fcc}^{e+1} = \boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e + \alpha_e \Delta\boldsymbol{\omega}_{fcc}^{e'}, \tag{2.15}$$

where $\boldsymbol{s}_e \doteq \Delta\boldsymbol{\omega}_{fcc}^e + \alpha_e \Delta\boldsymbol{\omega}_{fcc}^{e'}$; $\boldsymbol{d}_e \doteq \Delta\boldsymbol{\omega}_{fcc}^e = -[\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)^T \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e) + \mu_e \boldsymbol{I}]^{-1} \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)^T \mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e)$ is the normal LM step; $\Delta\boldsymbol{\omega}_{fcc}^{e'}$ is a line search for approximating the LM step, which is defined as

$$\begin{aligned} \Delta\boldsymbol{\omega}_{fcc}^{e'} = &- [\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e)^T \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e) + \mu_e' \boldsymbol{I}]^{-1} \\ &\cdot \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e)^T \mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_e), \end{aligned} \tag{2.16}$$

where $\mu_e' = \|\mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e)\|^\beta$, α_e is a parameter iterative updated as in (2.11) in Algorithm 2; $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e)$ is approximated by $\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)$; and μ_e' is approximated by μ_e for reducing the computational overhead. Then we can rewrite (2.16) as

$$\begin{aligned} \Delta\boldsymbol{\omega}_{fcc}^{e'} = &- [\boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)^T \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e) + \mu_e \boldsymbol{I}]^{-1} \\ &\cdot \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)^T \mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e + \Delta\boldsymbol{\omega}_{fcc}^e). \end{aligned} \tag{2.17}$$

In order to justify whether $\boldsymbol{s}_e$ is a good step or not, the trust region technique is used. The actual reduction R_a^e and the newly predicted reduction R_p^e at the eth iteration are defined in (2.18)

and (2.19), respectively.

$$R_a^e = \|\mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e)\|^2 - \|\mathcal{L}(L_2, \boldsymbol{\omega}_{fcc}^e + \boldsymbol{s}_e)\|^2 \tag{2.18}$$

$$\begin{aligned} R_p^e = &\|\mathcal{L}(\boldsymbol{\omega}_{fcc}^e)\|^2 - \|\mathcal{L}(\boldsymbol{\omega}_{fcc}^e) + \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)\boldsymbol{d}_e\|^2 \\ &+ \|\mathcal{L}(\boldsymbol{\omega}_{fcc}^e + \boldsymbol{d}_e)\|^2 \\ &- \|\mathcal{L}(\boldsymbol{\omega}_{fcc}^e + \boldsymbol{d}_e) + \alpha_e \boldsymbol{J}(\boldsymbol{\omega}_{fcc}^e)\Delta\boldsymbol{\omega}_{fcc}^{e'}\|^2. \end{aligned} \tag{2.19}$$

Their values are then compared by $r_e = R_a^e / R_p^e$, and the weights are updated according to the value of r_e as in (2.10) in Algorithm 2.

2.4 Evaluation with Real-world Datasets

Extensive experiments of load forecasting are conducted on two datasets at the system level and the residential level, respectively, to validate the performance of the proposed ensemble learning framework. The proposed framework is implemented with Keras 2.2.4, TensorFlow 2.0-beta, and Sklearn 0.20.0 in the Python 3.7 environment. The neural network for model fusion at level two is implemented using ADNBN coded by us using Matlab R2018a.

2.4.1 Datasets

Dataset Description

The following two public benchmark datasets are used for performance evaluation.

- The ISO-NE dataset [47]: This is a collection of hourly temperature and load data over 12 years from Jan. 1, 2007 to Dec. 31, 2018 in the New England area, including data for each of the eight zones (i.e., Connecticut-CT, Maine-ME, New Hampshire-NH, Rhode Island-RI, Vermont-VT, Massachusetts of NEM-NEMASS, Massachusetts of SEM-SEMASS, and Massachusetts of WC-WCMASS) and for the entire ISO-NE transmission system. Fig. 2.5

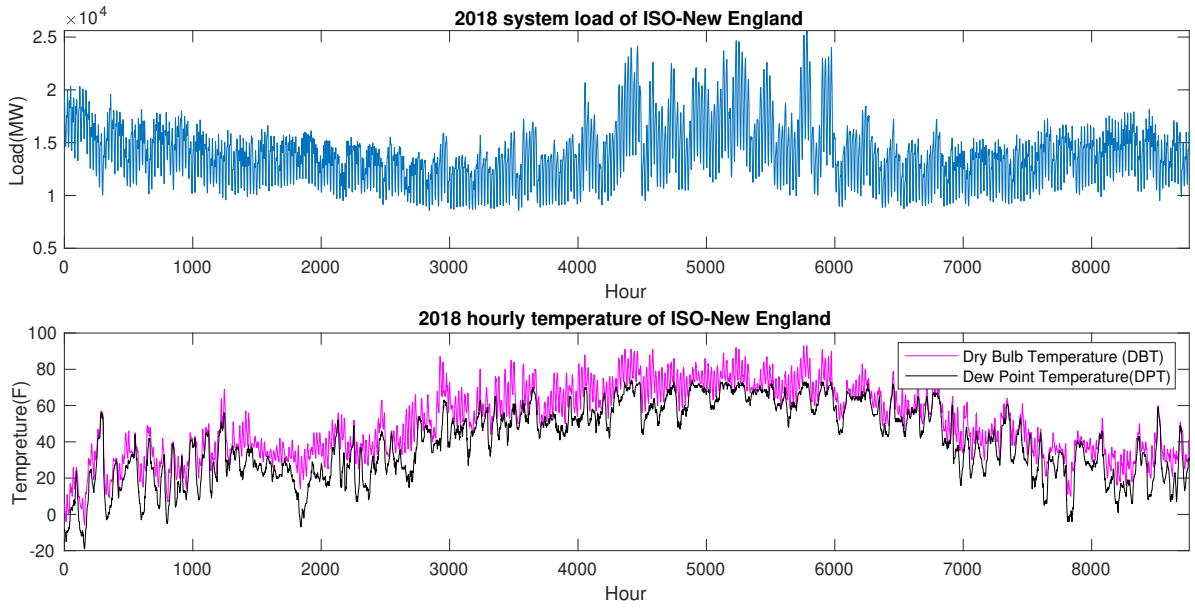


Figure 2.5: The 2018 overall system level load and temperature of the ISO-New England dataset.

presents the entire system level load and temperature data of the ISO-New England dataset in 2018. The load of each of the eight zones in 2018 is plotted in Fig. 2.6.

- The Residential Electricity Consumption dataset [48]: This is a collection of 370 clients' electricity consumption recorded for every 15 minutes during a period of three years from 2011 to 2014. Portuguese clients can be either residential or industrial consumers. Note that we only use the data for 320 clients, as the data for the remaining 50 clients are collected after 2011 (i.e., incomplete).

Preprocessing

A sliding window technique of P samples is implemented on historical time-series dataset during the training process. The period of P is divided into three parts, as shown in Fig. 2.2. The ratio of split is 2:1:1. For example, if hourly day-ahead load of Year the 2017 is predicted, the period P is set to 4 years. The data for one year from 2014 to 2015 partitioned to dataset $D1$, the data for

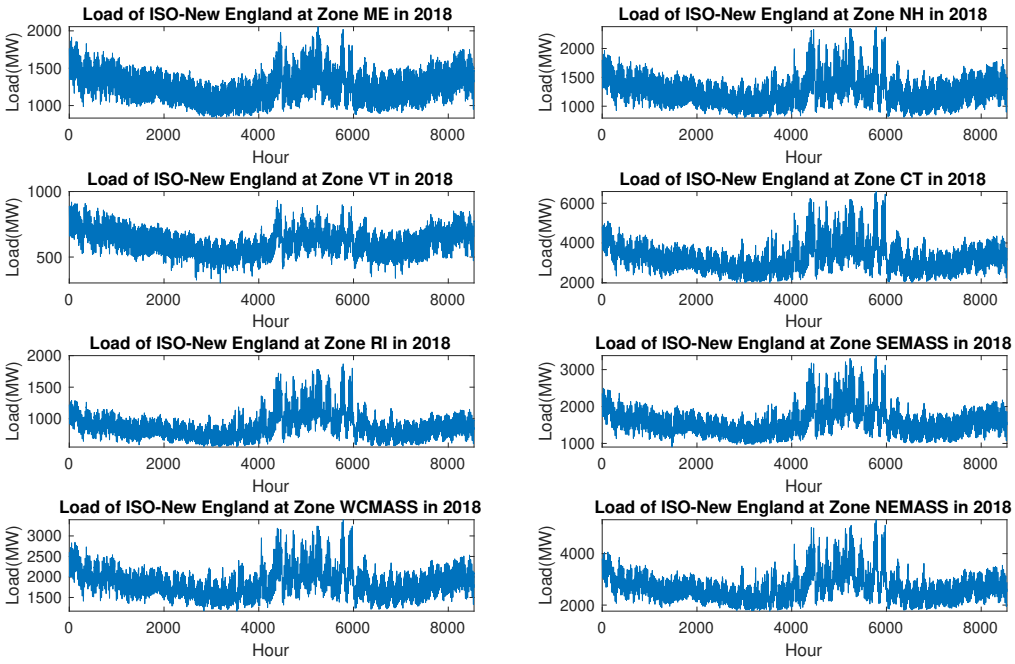


Figure 2.6: The individual load for each of the eight Zones in Year 2018 of the ISO-New England dataset.

2016 and 2017 become $D2$ and $D3$, respectively. When forecasting the load for the Year 2018, P is chosen from 2015 to 2018.

Normalization is applied in the preprocessing process. As shown in [49–51], normalization can not only speed up the convergence of training, but also reveal the true similarity between time series data. In order to prevent data snooping in time series prediction, which makes use of future information to enhance performance of forecast, only datasets $D1$ and $D2$ are normalized. In the testing set $D3$, new data generated by the first-level learner is restored from normalized form to the original form. The definition of normalization is

$$\boldsymbol{S}_{T;i}^{norm} = \frac{\boldsymbol{S}_{T;i} - \min(\boldsymbol{S}_{T;i})}{\max(\boldsymbol{S}_{T;i}) - \min(\boldsymbol{S}_{T;i})}, \tag{2.20}$$

where $\boldsymbol{S}_{T;i}^{norm}$ and $\boldsymbol{S}_{T;i}$ are the normalized and original form of data sample i in dataset $\boldsymbol{S}_T$, respectively.

Table 2.2: The Search Spaces of Algorithm Parameters

Algorithms	Parameters	Search Space
K-means++	number of clusters	[2:1:20]
Birch	number of clusters	[2:1:20]
DBSCAN	maximum distance between samples	[.5:.05:.8]
	minimum number of samples	[5:5:30]
HDBSCAN	minimum number of samples	[5:5:30]
LSTM	number of hidden neurons	[16,32,64,128]
	learning rate	[0.001,0.05,0.01]
	training epochs	[50:50:200]
FCC	number of hidden neurons	[2:1:11]
	training epochs	[50:50:150]
	activation function	[tanh,sigmoid,ReLU]

2.4.2 Experiments and Results

In our experiments, the grid search technique is applied for hyper-parameters tuning. The search space for the parameters in each machine learning algorithm is presented in Table 2.2.

System Level Prediction Performance

At the overall system level, short and mid term load forecasting are conducted on the ISO-NE dataset. The *first case* we examine is short-term forecasting, which predicts the load of the next day 24-hours ahead. In order to compare our method's performance with the existing cutting-edge technique, the system load in the Year 2010 and 2011 of ISO-NE are predicted individually, each using the three previous years' data as training and ensemble learning (see Section 2.4.1). We utilize the similar inputs as in [25]. Table 2.3 summarized the input of this case. For $feature_9$, the actual value of the temperature of the next day is used in all the schemes, based on the assumption that this information is available and the fact weather forecast is extremely accurate now-days.

Three state-of-the-art models proposed in [21, 22, 25] and the traditional LSTM recurrent neural network model are used as benchmarks for comparison with our proposed framework. The performance results in the form of mean absolute percentage error (MAPE) are shown in Table 2.4.

Table 2.3: Input Data and Output for Short-term Load Forecasting at Time h

Input	
$feature_1$	Load of the hth hour of the day that are 1, 2, 3, 4 and months prior to the next day
$feature_2$	Load of the hth hour of the day that are 1, 2, 3, 4 weeks prior to the next day
$feature_3$	Load of the hth hour of the day that are 1 days prior to the next day
$feature_4$	Load of the most recent 24 hours prior to the hth hour of the next day
$feature_5$	Temperature of the same hour as $feature_1$
$feature_6$	Temperature of the same hour as $feature_2$
$feature_7$	Temperature of same hour as $feature_3$
$feature_8$	Temperature of the hth hour of the next day
$feature_9$	Indicator (1,0) for season (winter, summer), weekend, and holiday
ℓ_h	Load at time h
Output	
$\hat{\ell}_{h+24}$	24 hours ahead load, i.e., $\tau = 24$

Table 2.4: Comparison of proposed model with Other Models using the ISO-NE Dataset for Years 2010 and 2011

	ISONE (SYS) 2010		ISONE (SYS) 2011	
Model's	MAPE	Number	MAPE	Number
ErrCorr modified [22]	1.75	-	1.98	-
ELM-PLSR [21]	**1.50**	-	1.80	-
DRN [25]	**1.50**	-	1.64	-
LSTM	1.58	-	**1.50**	-
K-means++-LSTM	1.30	8	1.32	8
DBSCAN-LSTM	1.37	6	1.34	7
BIRCH-LSTM	1.43	9	1.34	11
HDBSCAN-LSTM	**1.29**	15	**1.30**	13

The number of first-level learners in our proposed module is presented in the second column for each year as well. The table shows that the four variants of our proposed framework all outperform the four benchmark schemes. An average reduction in MAPE of 10.17% in the Year 2010 and 11.67% in the Year 2011 are achieved over the four baseline schemes.

Table 2.5: Individual HDBSCAN based Model Results and the Ensemble Method Improvement for the System Level Load in Years 2010 and 2011

	ISONE (SYS) 2010	ISONE (SYS) 2011
Model	MAPE	MAPE
1	1.679	2.589
2	1.905	1.600
3	2.415	1.472
4	2.171	**1.317**
5	1.778	2.108
6	1.798	2.347
7	2.196	1.396
8	1.460	1.378
9	1.518	2.808
10	1.373	1.351
11	1.506	1.386
12	**1.307**	†9.685
13	1.448	1.377
14	1.757	-
15	2.303	-
Combined Model	**1.291**	**1.299**

We also find that the HDSCAN based approach outperforms the other variants of our framework. To illustrate the efficacy of ensemble learning, we also present the performance of the first-level and second-level learners in Table 2.5. The table shows that there are 15 and 13 base LSTM models f Years 2010 and 2011, respectively. That is, for each year, the dataset $D1$ is partitioned into 15 and 13 groups, respectively, for training the first-level LSTM models. The table also shows that the second-level learning by the FCC neural network effectively further reduces the MAPE. Compared with the MAPEs in the first-level learner, the FCC achieves an average improvement in MAPE of 21.59% and 25.60% for the Year 2010 and 2011, respectively. To visualize the performance results, the forecast results of the last two weeks in 2011 predicted by the HDBSCAN based LSTM model are plotted along with the ground truth in Fig. 2.7. It can be seen that the forecast curve matches the ground truth tightly.

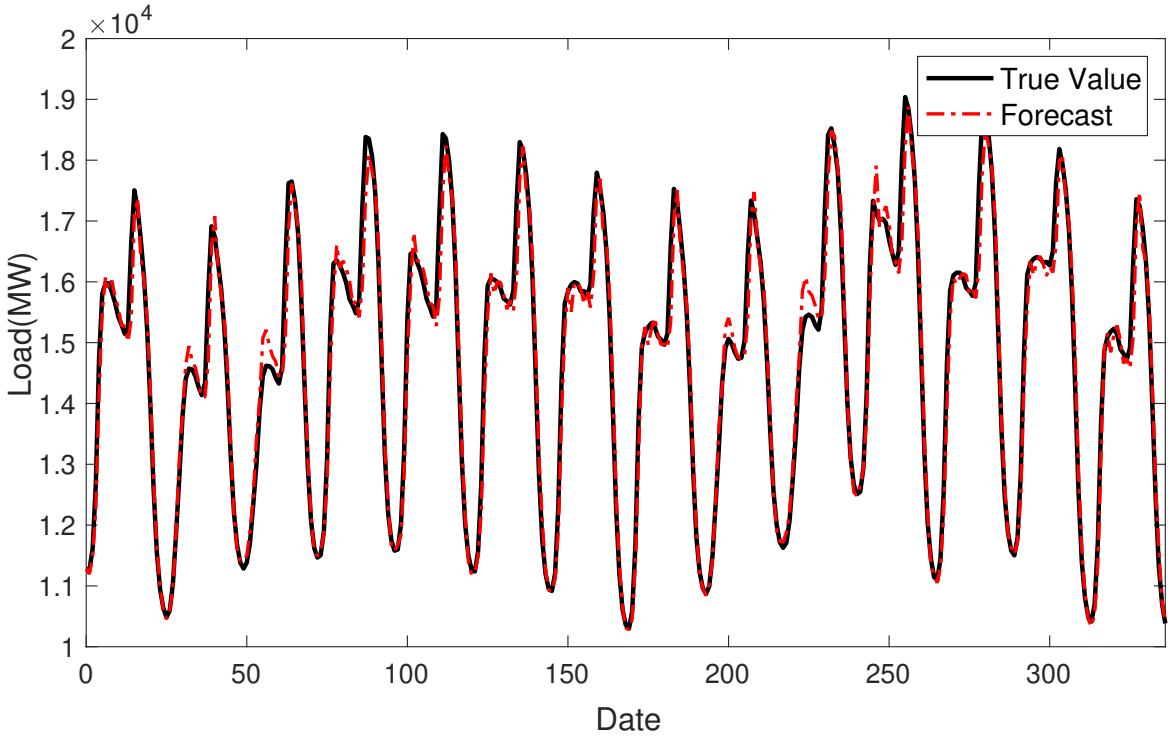


Figure 2.7: System load forecast results for the last two weeks of 2011 on the ISO-NE dataset using the HDBSCAN-LSTM model.

In the 2011 prediction results, the performance of model 12 is marked with a symbol "†," which indicates the worst score MAPE among all the 13 LSTM models. We carefully examine this case and plot the clustering result for this prediction in Fig. 2.8. It can be seen that each of the other 12 clusters has a sufficient number of samples, while only 69 samples are grouped into the 12th cluster. This level-one learner (LSTM model 12) is trained with a very small dataset. As a result, it has a comparatively weak ability of generalization. It achieves the worst performance as the features extracted by this model are not general enough and are only suitable and specific to the sample dataset (Cluster 12).

We further explore the effect of the number of hidden neurons in the second level of learning on the prediction. Table 2.6 shows the average training and testing error (i.e., Normalized Root Mean Square Error) learned by the HDBSCAN based LSTM model with different numbers of hidden neurons. In each trial, the neural network with the same number of hidden neurons is

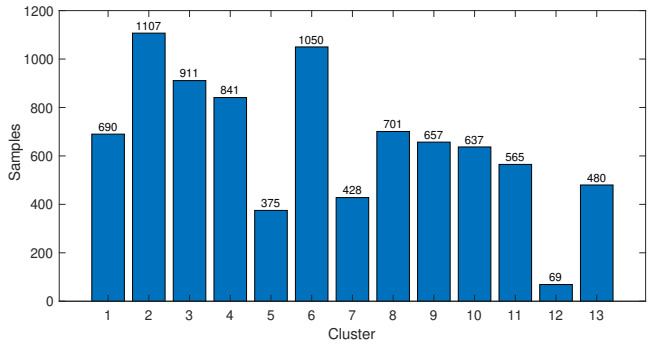


Figure 2.8: Sample distribution of the HDBSCAN model for system level load prediction in Year 2011.

Table 2.6: The Effect of the Number of Hidden Neurons on the Training Process

Number of	ISONE (SYS) 2010 Average NRMSE		ISONE (SYS) 2011 Average NRMSE	
Hidden Neurons	Training	Testing	Training	Testing
2	0.0203	0.0212	0.0191	0.0216
3	0.0209	0.0218	0.0185	0.0209
4	0.0215	0.0223	0.0183	0.0207
5	0.0203	0.0212	0.0186	0.0209
6	0.0206	0.0214	0.0187	0.0211
7	0.0207	0.0215	0.0184	0.0206
8	**0.0192**	**0.0201**	0.0182	0.0205
9	0.0216	0.0224	0.0188	0.0212
10	0.0208	0.0217	0.0180	0.0201
11	0.0198	0.0208	**0.0179**	**0.0201**

trained 100 times, and the average training and testing errors are presented in the table. As shown in the table, increasing the number of hidden neurons does not guarantee to reduce the training and testing errors. The minimum training and testing errors are achieved with 8 hidden neurons for ISONE (SYS) 2010 and with 11 hidden neurons for ISONE (SYS) 2011. Finding a proper parameter (i.e., the number of hidden neurons) is vital for the training process. Thus, the grid search technique is applied in our proposed framework.

As mentioned in Section 2.2.4, the FCC neural network's hidden neurons are activated by the $\tanh(\cdot)$ function. In order to explain why we choose this activation function, we compare the

performance (i.e., the learning curve) of different activation functions. The model is trained by $\tanh(\cdot)$, $\text{sigmoid}(\cdot)$, and $\text{ReLU}(\cdot)$, respectively, with the same input and neural network structure. This experiment is implemented with the HDBSCAN-LSTM model, which has 3 hidden neurons, using Year 2010 data. Fig. 2.9 presents the learning curves, training and testing errors, as well as training and testing time. It indicates that the $\tanh(\cdot)$ and $\text{sigmoid}(\cdot)$ activation functions are more stable than the $\text{ReLU}(\cdot)$ function. Although the $\text{sigmoid}(\cdot)$ function takes less time for training, the $\tanh(\cdot)$ function achieves a slightly better performance on reducing the training and testing error.

The *second case* we examine is to forecast week-ahead power load on weekends (i.e., for Saturday and Sunday) at both the zone level and system level for the Year 2018. The data from 2015 to 2017 are used for training the models. The output is the weekend's hourly load values. In this task, we only use historical temperature data as a feature. The current temperature (i.e., at $t + \tau$) is not used in this forecast, which is different from the previous case. This is because in practice, weekly ahead weather forecast is not as precise as day-ahead weather forecast. In order to mimic the actual situation in forecasting, we only use the feature information that is available at the forecasting time instance in this study (i.e., no future information is available). Therefore, the input of this case is weekly lagged temperature and power load time series data.

The evaluation results of both Root Mean Square Error (RMSE) and MAPE are summarized in Table 2.7 (testing errors) and Table 2.8 (training errors), for both the overall system-level load prediction (the first row) and that for each of the zones in the New England area (the remaining eight rows). We compare the four variants of the proposed framework with the basic LSTM model using the same input. Apparently, our proposed framework performances better than the traditional LSTM model. The HDBSCAN based model consistently outperforms all the other models in this experiment.

Residential Level Prediction Performance

We next study the load forecasting problem for individual clients using the proposed ensemble learning model on the Residential Electricity Consumption dataset [48]. The electricity load data

Table 2.7: Comparison of the Four Variants of the Proposed Model with the Basic LSTM model on the ISO-NE Dataset for Weekly Ahead Hourly Load Forecast on Weekend Days in Year 2018: Testing Errors

	LSTM		Kmeans++-LSTM		DBSCAN-LSTM		BRICH-LSTM		HDBSCAN-LSTM	
ZONE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ISONE(SYS)	1279.639	6.396	1172.054	5.987	1138.088	5.829	1143.666	5.878	**754.019**	**4.369**
CT	332.453	7.008	336.177	7.274	324.761	6.881	332.367	6.821	**187.77**	**4.610**
NH	108.098	5.822	107.734	5.779	105.366	5.671	108.381	5.812	**75.092**	**4.837**
ME	79.852	4.487	78.204	4.427	76.080	4.327	79.336	4.485	**60.137**	**3.680**
RI	92.787	6.411	89.279	6.566	85.895	6.243	87.864	6.606	**48.600**	**4.431**
VT	58.264	7.614	50.924	6.172	53.261	6.554	53.478	6.783	**46.746**	**5.829**
SEWASS	178.423	7.330	168.339	7.210	169.527	7.352	166.968	7.168	**113.50**	**5.512**
WCMASS	173.362	6.421	158.812	6.114	164.382	6.223	163.230	6.327	**129.867**	**5.687**
NEWASS	290.23	6.747	278.474	7.071	259.557	6.393	262.456	6.535	**177.01**	**5.059**

Table 2.8: Comparison of the Four Variants of the Proposed Model with the Basic LSTM model on the ISO-NE Dataset for Weekly Ahead Hourly Load Forecast on Weekend Days in Year 2018: Training Errors

	LSTM		Kmeans++-LSTM		DBSCAN-LSTM		BRICH-LSTM		HDBSCAN-LSTM	
ZONE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
ISONE(SYS)	1203.829	6.049	652.820	4.059	637.553	3.990	605.226	3.789	576.435	3.412
CT	318.441	6.335	168.854	4.373	187.350	4.745	176.095	4.699	174.893	4.383
NH	96.405	5.175	59.860	3.901	60.260	3.963	58.533	3.836	57.924	3.567
ME	59.889	3.734	47.506	3.041	45.875	2.895	44.091	2.792	50.535	2.973
RI	77.740	5.381	34.034	3.263	35.478	3.461	37.404	3.617	36.910	3.274
VT	52.537	6.644	50.396	6.019	52.812	6.403	53.261	6.554	45.881	5.801
SEWASS	147.80	5.864	94.874	4.951	103.588	5.225	92.165	5.001	91.264	4.960
WCMASS	149.845	5.502	108.455	4.609	112.473	4.824	97.647	4.406	94.990	4.436
NEWASS	142.367	3.938	108.531	3.489	116.474	3.669	94.975	3.020	93.519	3.097

is aggregated from every 15 minutes to one hour. The aggregated dataset is spitted into three parts as described in Section 2.4.1. Then the 320 clients are classified into several groups using the HDBSCAN clustering algorithm based on the data of the first three months in 2012. Fig. 2.10 presents the clustering result. It shows that the consumers are grouped into five clusters. In Figs. 2.10(a)-(e), each curve represents the normalized load of a client, while Fig. 2.10(f) shows the number of clients in each cluster. We find that each cluster is visually different. For example, the load curves in Cluster 4 are all relatively flat and are close to 0.5, the load curves in Clusters 2 and 3

are serrated and ranging from 0.2 to 1, and the load curves in Cluster 5 exhibit an obvious daily pattern (indicating these are residential clients).

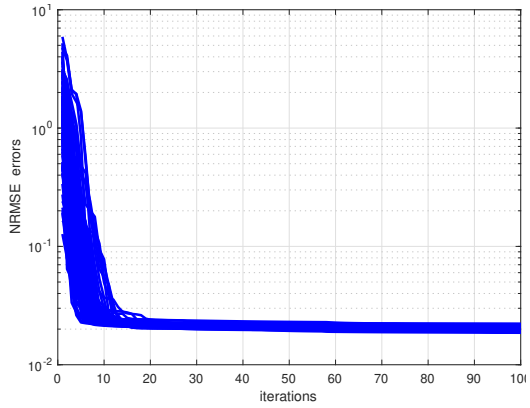
In each cluster, we randomly select a client to predict its load. Thus, for each client, historical load time series right before time step t is used as the training set, where the window size W is set to 168 (the number of hours in seven days). The dimension of input $m \times W$ at time t is 1×168 since we only use the historical load data (i.e., $m = 1$). The output is the predicted load for a future time $t + \tau$.

We use the traditional LSTM model as a baseline scheme. Table 2.9 summarizes the evaluation results in the form of MAPE for the five chosen clients (one from each cluster as shown in Fig. 2.10). The horizon τ is set to be 1, 12, and 24, respectively, which means we predict hour ahead, half-day ahead, and day-ahead load values for the five selected clients. Compared with LSTM, the proposed ensemble learning models achieve a much higher precision in this experiment. For example, for Client 4, the BIRCH MAPEs are 23.35%, 26.43%, and 33.64% of the corresponding LSTM MAPEs for $\tau = 1$, $\tau = 12$, and $\tau = 24$, respectively. The best result for different clients and horizon τ is different. However, the proposed ensemble learning models all achieve the best performance. Among all the results, BIRCH and HDBSCAN based models perform better, which achieve the lowest errors comparing to others. Considering HDBSCAN is an improved algorithm of DBSCAN, density-based algorithm HDBSCAN and hierarchical algorithm BIRCH are superior to partitioning algorithm K-means++, which suffers from outliers or noise, in this case.

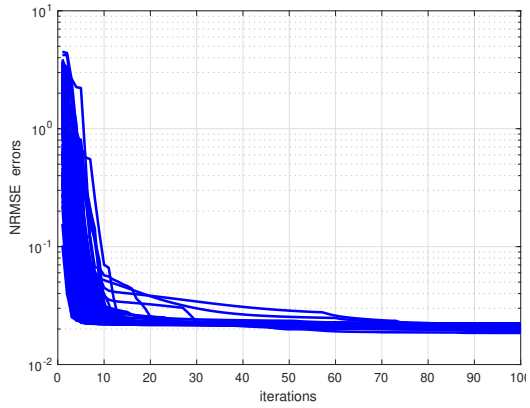
For all models, the MAPE of Client 1 is relatively high, while the MAPE of all the other clients are all below 21. From Fig. 2.10, we can see that in cluster one, there is no obvious trend for this group of data, which might explain why this group of data is difficult to forecast. It is extremely challenging to accurately predict every client's load due to different lifestyles or activities. Classifying the clients and predict load by the group is quite feasible as shown in this experiment.

Table 2.9: MAPE Comparison of the Clustering Based Model and the Basic LSTM Model on Residential Data

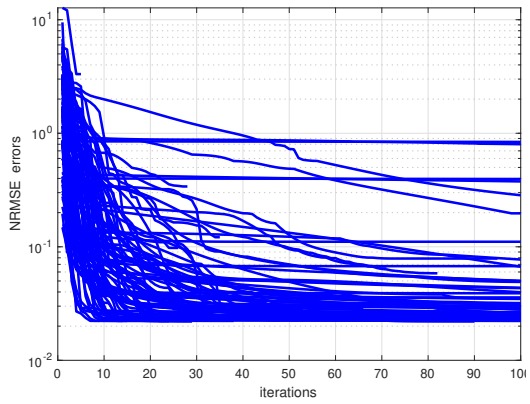
Resident	K-means++			BIRCH			DBSCAN			HDBSCAN			LSTM		
	$\tau = 1$	$\tau = 12$	$\tau = 24$	$\tau = 1$	$\tau = 12$	$\tau = 24$	$\tau = 1$	$\tau = 12$	$\tau = 24$	$\tau = 1$	$\tau = 12$	$\tau = 24$	$\tau = 1$	$\tau = 12$	$\tau = 24$
1	**18.15**	43.59	**43.51**	18.16	55.28	46.89	23.74	45.36	44.54	19.33	**42.21**	44.32	36.15	42.88	48.85
2	3.79	7.70	8.34	3.70	7.76	8.45	**3.66**	**6.78**	8.33	3.75	7.33	**7.96**	14.51	10.27	9.78
3	11.90	17.06	15.81	15.54	**15.15**	15.24	12.72	15.4 1	16.33	**11.21**	15.24	**15.12**	22.23	24.50	23.12
4	9.31	16.41	13.92	**8.53**	**10.19**	**12.90**	12.09	12.29	13.11	8.83	12.55	12.94	36.53	38.55	38.34
5	12.63	16.27	14.76	14.53	20.52	**14.40**	12.57	15.71	14.90	**11.24**	**14.91**	15.22	27.74	23.58	24.19



(a) tanh activation function: average training error (NRMSE) is 0.0210 ± 0.0011, average testing error (NRMSE) is 0.0219 ± 0.0011, and average training time is 42.0660 seconds.



(b) sigmoid activation function: average training error (NRMSE) is 0.0214 ± 0.0010, average testing error (NRMSE) is 0.0222 ± 0.0009, ad average training time is 38.1347 seconds.



(c) ReLU activation function: average training error (NRMSE) is 0.1347 ± 0.3834, average testing error (NRMSE) is 0.1331 ± 0.3698, and average training time is 42.8419 seconds.

Figure 2.9: Learning curves of the ensemble neural networks (FCC) with different activation functions.

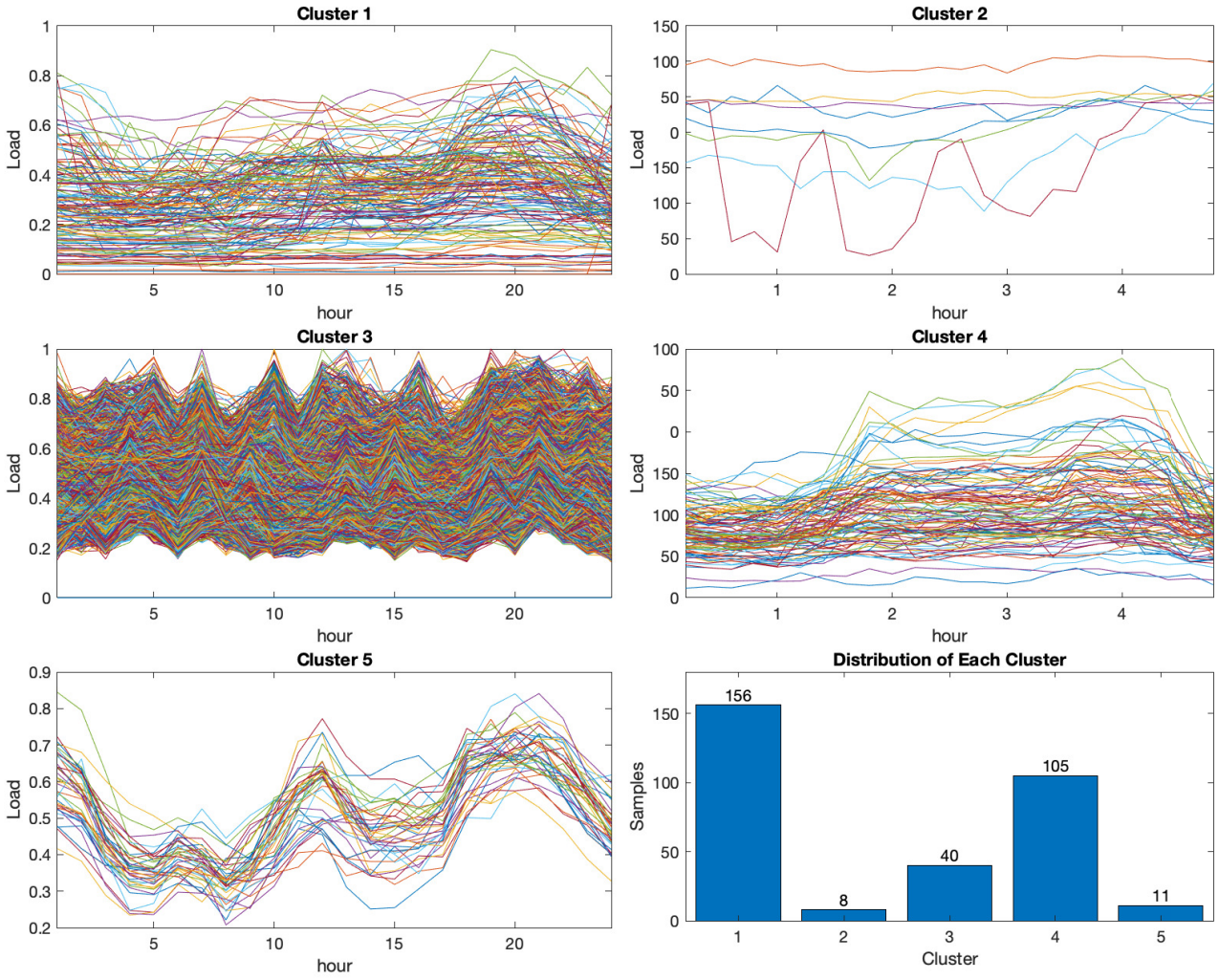


Figure 2.10: Distribution of classified residents based on the DBSCAN clustering algorithm.

Chapter 3

Pre-trained Models for Non-intrusive Appliance Load Monitoring

3.1 Non-Intrusive Load Monitoring

With extremely low latency, high data rate, and significant improvement of quality of service (QoS), the 5G and beyond wireless networks offer considerable benefits in many fields. However, the tremendous energy usage, estimated to be 10 times more than the existing 4G networks, has raised great concerns [52]. Inspired by the advances in green communications and networking (GCN), which aims to Send More Information bits with Less Energy (SMILE), energy-efficient techniques, such as BS switching [53–55], offline power allocation and online data scheduling, as well as sustainable energy powered base stations (BSs) have been developed to reduce the energy usage and boost network capacity [56]. GCN has a close interaction with the power grid. On one hand, for retailers, communication networks collect data and information from the power grid components, which can be analyzed and used to control the power system for real-time pricing, demand response, and protection [18]. On the other hand, for consumers, networks construct communication paths that integrate smart meters, home appliances, and renewable energy sources for Home Energy Management Systems (HEMS) [19, 20].

Among the HEMS applications, Non-Intrusive Load Monitoring (NILM) has been recognized as an essential component. The goal is to estimate each individual appliance's power consumption

from the aggregated smart meter data; it is non-intrusive since only the aggregated power consumption is needed [57, 58]. The most crucial advantage of NILM is its nonintrusive character. Comparing to intrusive approaches, NILM can provide appliances energy usage information without sub-meter installation, which is expensive, hard to upgrade, and causes data privacy concerns [59]. With the NILM results, homeowners can enjoy the benefits of optimizing energy assets to achieve energy savings. It was reported in [60] that feedback on power usage stimulates energy savings ranging from 1.1% to over 20%. As residential consumers have increasingly adopted more electric vehicles (EVs) and home solar systems, instant information about their energy consumption and generation will help to optimize their energy utilization. Another benefit for consumers is equipment malefaction detection. NILM provides feedback when an appliance, e.g., air-conditioner or refrigerator, consumes more energy than expected with anomaly detection algorithms without the need for sub-meter level data. For retailers, NILM can also help to improve their energy management (power system scheduling and planning). Provided with customer's consumption behavior from NILM, retailers can provide customized services, such as offering energy-saving tips (i.e., informing consumers to lower power consumption when the wholesale market prices are high) and enabling different billing methods (static or dynamic), to improve customer satisfaction [61].

Although NILM brings about great benefits, it faces many challenges as well. The most successful approach to NILM, so far, is deep learning (DL), which achieves the state-of-the-art performance. However, it requires a large amount of labeled data to train the DL model. For NILM, this requires electrical submetering in the houses, which is to use additional electricity monitors to record the usage of individual appliances in the house, and thus incurs additional costs. Furthermore, as people are more concerned about their privacy, the active power data used for training the NILM model is hard to obtain. Moreover, as most data-driven models, the DL approach requires extensive computation. It would be desirable to eliminate the need to train a model every time it is used for a new house. Therefore, for practical deployment of NIML solutions, it is critical to develop DL models that are *generalizable*, such that we can train the model with data collected from a small number of submetered houses, and then easily apply the

trained model to other houses without submetering. Such generalizable DL models are also useful to deal with houses with different appliances, different residents' usage behavior, and various aging degrees of circuits [62].

In this chapter, we investigate the problem of pre-trained DL models for NILM, which is a promising means to address the above problems. With this approach, a base DL model is first trained with a larger dataset. When applied to a new environment, the base model is first fine-tuned with a small amount of new data, and then the fine-tuned model is used for inference in the new environment. On one hand, we do not need to train a new model from scratch for an unknown house. On the other hand, pre-trained models are able to quickly adapt to new tasks with few-shot learning, as pre-trained parameters outperform random initialization for deep neural networks [63]. Therefore, such models can not only save considerable computation in training and reduce the dependency on large amounts of data, but also achieve excellent performance in real-time when it is allowed to use new data to update the parameters.

In light of these, we propose a model-agnostic meta-learning (MAML) based approach and an ensemble learning based approach for the NILM problem in this chapter. Our approaches are inspired by two of the most successful Natural Language Processing (NLP) pre-trained transformer models BERT [64] and GPT-3 [65]. Ensemble learning (BERT) and meta-learning (GPT-3) are the two effective solutions toward improving the pre-training language model's adaptability. We propose these two approaches to deal with the transferability of the NILM problem. Both methods obtain the pre-trained models using one dataset, and then fine-tunes their parameters using a small amount of new data when applied for inference with another dataset. To the best of our knowledge, this is the first work that applies meta-learning and ensemble learning for generalizable models to the NILM problem. We develop both models and evaluate their performance with two real-world datasets, using one dataset to pre-train the models and the other dataset to fine-tune the models and test their generalization performance. Our experiments validate the superior transferability of the proposed models for the NILM problem, which both outperform the state-of-the-art DL based approach and the transfer learning based approach [1]. We also find that the proposed models

are effective in overcoming the negative transfer problem. The proposed models require greatly reduced amount of data and computation for real-world deployment, which lead to energy savings and is inline with the goals of GCN.

We organize the remainder of this chapter as follows. Related work is introduced in Section 3.2. In Section 3.3, we formulate the NILM problem and introduces several solution approaches. In Section 3.4, we present the two proposed methods. We present the datasets and experiment setup in Section 3.5, and our experimental validation of the proposed models in Section 3.6. Section 3.7 concludes this chapter.

3.2 Related Works

3.2.1 The NILM Problem and Existing Solutions

The wide deployment of smart meters has triggered great interest in NILM, which is to estimate the power consumption of a target appliance from the aggregate meter readings of the entire house. Many algorithms have been developed to address the NILM problem. For example, the Additive Factorial Hidden Markov Model (AFHMM) and its variants have been used in many existing schemes [66–70]. The Graph Signal Processing (GSP) based method has also been shown to be quite effective [71, 72]. Other traditional machine learning approaches, such as Support Vector Machine (SVM) [73], Decision Trees [74], the hybrid classification method [75], k-nearest neighbors (k-NN) [76], and so forth, have been applied to solve the NILM problem as well. Interested readers are referred to the detailed reviews in [59, 77]. Note that such works only focus on training and inference with the same dataset, rather than the generalization problem.

Motivated by the success of deep learning in other fields, there has been great interest in applying deep learning to solve the NILM problem [78]. Convolutional Neural Networks (CNNs) models have been adopted in [79–81] to extract the temporal features from time series of aggregate electricity consumption data. In [82], Long Short-Term Memory (LSTM) or its equivalent Gated Recurrent Units (GRUs) models have been leveraged to capture the long and short-term patterns

of state signatures of different appliances, which belong to the class of Recurrent Neural Networks (RNNs). De-noising auto-encoder has also been applied for noise reduction to better estimate the appliance profile [83].

3.2.2 Pre-trained Models

Recent work has shown that by pre-training a deep neural network on a large corpus of data, followed by fine-tuning when applied to a specific task, the model's performance on the target task can be effectively improved. This approach has been successfully applied in computer vision, speech recognition, and especially in NLP.

A pre-trained hidden Markov model for large-vocabulary speech recognition was proposed in [63]. The authors showed that the pre-trained model was robust and achieved good initialization of weights when training deep neural networks. For computer vision, the authors in [84, 85] explored image feature transferability of CNNs and found that the pre-trained model could boost the generalization performance to new image classification tasks. Recently, pre-trained models have drawn considerable attention in NLP. For example, ELMo (Embeddings from Language Models) [86] is a feature-based NLP pre-training approach, which combines individual feature extract LSTMs to improve the overall task performance. The pre-trained transformer language model BERT [64] can effectively handle multiple NLP tasks, after being fine-tuned directly without the need for task-specific architectures. In 2020, OpenAI launched GPT-3, a gigantic deep neural network with 175 billion parameters [65], to tackle task-agnostic NLP problems without needing any gradient updates or fine-tuning. Motivated by the success of pre-trained models in other fields, we investigate how to apply it to solve the NILM problem in this chapter.

3.2.3 Pre-trained Models for NILM

There has been very few existing works on pre-trained models for NILM. Most of the prior works trained and tested their models using the dataset from the same house, by partitioning the same dataset into a training set and a testing set. The generalization performance of the models has not

been verified. In [83, 87], the authors considered transferability across houses included the same dataset, i.e., testing a model trained by one house and on an untrained house, which both belonged to the same dataset. In [1, 62], houses from different datasets were used, where a model was pre-trained on a large dataset and then its transferability and generalization performance was verified through another domain. The difference between them was that the work [62] tested its model without any parameter updating, which is known as *zero-shot learning*. Usually transferring a model between two different datasets could lead to poor performance. In [1], the pre-trained model was fine-tuned by using data from the other dataset (i.e., few-shot fine-tuning). The limitation of [1] was that the fine-tuned model's performance was sometimes worse than the zero-shot models. This was because the data used in fine-tuning was quite different from that of the tested house, which led to *negative transfer*. The generative adversarial networks (GANs) are used as the pre-trained model in [88, 89]. By minimizing the statistical distance between source and target domains in the feature space, the authors in [88] overcame the drawback that the shared parameters of the pre-trained model are sensitive to the similarity between different domains. In [89], the joint adaptation loss was further introduced by adapting both the feature and the label distribution discrepancy, which improved the performance of GANs.

Another approach of using pre-trained models for NILM is to train a model on visual recognition tasks and the transfer the image feature extractor to the appliance recognition task. To bridge these two unrelated domains, i.e., computer vision and NILM, the authors in [90] introduced the concept of a load signature, i.e., the voltage-current (V-I) trajectory, to enable transfer learning. Since the features extracted from the NILM data is usually quite different from real-world images, it is challenging to verify the model's robustness to domain shits (i.e., from real images to power consumption data).

3.3 Problem Statement and Approaches

In this section, we first present the mathematical formulation for the Non-Intrusive Load Monitoring (NILM) problem. We will then introduce the conventional supervised machine learning (ML) and pre-training approaches (i.e., transfer learning and meta-learning) to solve the problem.

3.3.1 The NILM Problem

Consider a house that contains J appliances that consume electricity. The aggregated power consumption of the house, as measured by a smart meter, is given by

$$x(t) = \sum_{j=1}^{J} y_j(t) + e(t), \tag{3.1}$$

where $x(t)$ is the aggregated power consumption, $y_j(t)$ is the jth appliance's power consumption, and $e(t)$ is the measurement noise at time t. Given measurement of the total power consumption over a time period T, i.e., $\tilde{\mathbf{x}} = (x(1), x(2), ..., x(T))$, the goal of NILM is to estimate the individual appliance's power consumption trace for the same period T, i.e., $\tilde{\mathbf{y}}_j = (y_j(1), y_j(2), ..., y_j(T))$, for $j = 1, 2, ..., J$.

Supervised ML has been applied to solve the NILM problem, as reviewed in Section 3.2, which is to train a model with observed pairs of $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j)$ (i.e., the labeled training set) to estimate (i.e., learn) an approximate function $f_{\boldsymbol{\theta}}(\cdot)$ over a parameter set $\boldsymbol{\theta}$ with a learning algorithm, which represents the relationship between $\mathbf{y}_j$ and $\mathbf{x}$ by

$$\mathbf{y}_j = f_{\boldsymbol{\theta}}(\mathbf{x}). \tag{3.2}$$

Various learning models can be applied to solve the NILM problem. For example, the conventional ML approach utilizes a single learning algorithm to learn the function $f_{\boldsymbol{\theta}}(\cdot)$. On the other hand, transfer learning leverages a base learner to learn the function, and then utilizes new data to adapt to a new domain. Meta-learning, known as *learning to learn*, incorporates several learning

episodes to induce the learning algorithm itself. In the remainder of this section, we describe how to solve the NILM problem with these ML approaches from an optimization perspective. We will use two separate load monitoring datasets, i.e., a source dataset $\mathcal{S}$ and a target dataset $\mathcal{T}$, in the following discussions. Both datasets contain the aggregated power consumption data as well as the consumption data of individual appliances (as labels).

3.3.2 Conventional Machine Learning Approach

To solve the NILM problem with a conventional ML approach, only one dataset $\mathcal{S}$ is used. This dataset is split into two parts, i.e., a training set $\mathcal{S}^{tr}$ and a testing set $\mathcal{S}^{ts}$. The ML model is trained with the training set $\mathcal{S}^{tr}$ to determine its parameters $\boldsymbol{\theta}$. The trained model is then tested on the separate testing set $\mathcal{S}^{ts}$.

The goal of the training process is to minimize a loss function $\mathcal{L}$, give by

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \ \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}^{tr}\right). \tag{3.3}$$

The model parameters $\boldsymbol{\theta}$ are usually updated with the gradient descent (GD) method as follows.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}^{tr}\right), \tag{3.4}$$

where η is the learning step size, and $\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}^{tr}\right)$ is the gradient of the loss function with respect to $\boldsymbol{\theta}$. When the model is well trained, its performance will be evaluated using the testing set $\mathcal{S}^{ts}$. Such a process is illustrated by the graphical model given in Fig. 3.1.

The conventional supervised ML approach to the NILM problem usually focuses only on a single dataset $\mathcal{S}$. The model parameters are optimized with respect to this dataset. Usually the trained model does not generalize well to an untrained dataset $\mathcal{T}$ (i.e., a new domain).

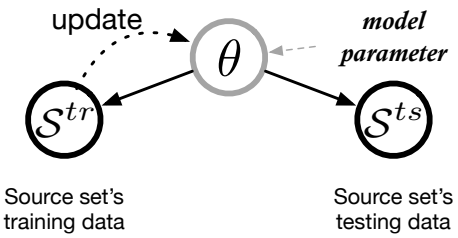


Figure 3.1: Conventional supervised ML approach: the model parameters are updated based on a single task training dataset $\mathcal{S}^{tr}$ and tested on a testing set $\mathcal{S}^{ts}$.

3.3.3 Transfer Learning Approach

In transfer learning, both a source dataset $\mathcal{S}$ and a target dataset $\mathcal{T}$ are used. The goal is to adapt the pre-trained model learned from the source dataset $\mathcal{S}$ to the target dataset $\mathcal{T}$.

The procedure is illustrated in Fig. 3.2. First, we create the training dataset $\mathcal{S}^{tr}$ from $\mathcal{S}$ to pre-train the model. The pre-training problem can be defined as

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \ \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}^{tr}\right). \tag{3.5}$$

With the gradient descent (GD) method, the model parameters $\boldsymbol{\theta}$ are updated using the training set $\mathcal{S}^{tr}$ as:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}^{tr}\right). \tag{3.6}$$

In the testing phase, the target dataset $\mathcal{T}$ is split into a fine-tuning set $\mathcal{T}^{tr}$ and a testing set $\mathcal{T}^{ts}$. We fine-tune the pre-trained model using the fine-tuning set $\mathcal{T}^{tr}$ for purpose of *domain adaptation*, where the parameters are updated as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{T}^{tr}\right). \tag{3.7}$$

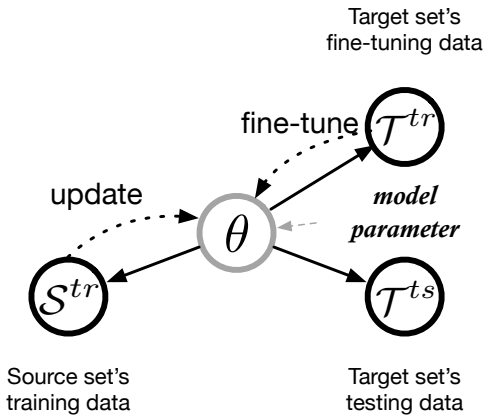


Figure 3.2: Transfer learning approach: the model parameters are updated based on the training set $\mathcal{S}^{tr}$, fine-tuned on set $\mathcal{T}^{tr}$, and tested on set $\mathcal{T}^{ts}$.

Then the fine-tuned model is tested on the testing set $\mathcal{T}^{ts}$. During the fine-tuning stage in transfer learning, most existing works freeze the parameters in most of the layers, except the last fully-connected layer. The fine-tuned model is obtained by training the parameters of the last fully-connected layer using the new task's data $\mathcal{T}^{tr}$.

In our prior work [20], we developed an ensemble learning model for load forecasting in urban power systems, which includes multiple long short-term memory (LSTM) based first-level learners and a Fully Connected Cascade (FCC) neural network as the second-level learner. In this chapter, we propose an ensemble learning based transfer learning approach to solve the NILM problem. The proposed model will be presented in Section 3.4.

3.3.4 Meta-learning Approach

Meta-learning, a.k.a. learning to learn, is inspired by human's quickly learning new things with only a few examples. By applying automatic learning algorithms to metadata, it induces the learning algorithm itself. The goal is to enable an intelligent agent (i.e., model) learn and adapt quickly from few-shot of examples, and is able to keep adjusting as more data are coming in [91].

In general, meta-learning can be seen as training a general model that can generalize across different tasks or datasets. Here, we define a single task or dataset as $\mathcal{S}_i$, which is sampled from the source set $\mathcal{S}$. We sample the source set $\mathcal{S}$ for N times to obtain N tasks. Each task $\mathcal{S}_i$ is split into

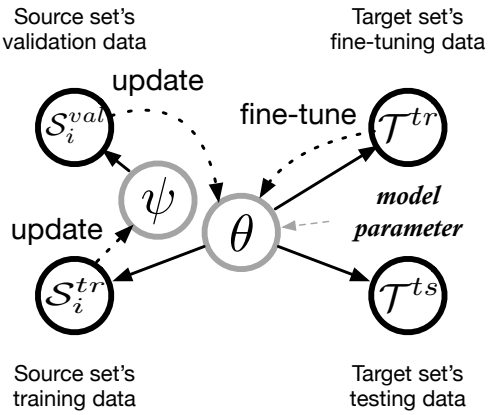


Figure 3.3: Meta learning approach: The model parameters are updated based on two meta training datasets $\mathcal{S}^{tr}$ and $\mathcal{S}^{val}$, fine-tuned on set $\mathcal{T}^{tr}$, and tested on set $\mathcal{T}^{ts}$.

a training set $\mathcal{S}_i^{tr}$ and a validation set $\mathcal{S}_i^{val}$. In meta-training, the model (meta-learner) shares the parameters $\boldsymbol{\theta}$, which will be updated with each task's loss. The average of parameters optimized by each task (i.e., the base learners represented by parameters $\boldsymbol{\psi}$) will update the meta-learner at last. This way, the meta-learner will fit all tasks at the same time, akin to cross-validation. The target set $\mathcal{T}$ will be partitioned into a fine-tuning set $\mathcal{T}^{tr}$ and a testing set $\mathcal{T}^{ts}$. The pre-trained meta-learner will be fine-tuned on $\mathcal{T}^{tr}$ and tested on $\mathcal{T}^{ts}$.

In this chapter, we will adopt Model-Agnostic Meta-Learning (MAML) [91], which is an optimization scheme, to solve the NILM problem. Detailed implementation of the proposed model will be described in the next section.

3.4 Proposed Approaches

In this section, we present two approaches to the NILM problem, focusing on the generalization of the models. The first model is a meta-learning based approach, i.e., MAML, that relies on fine-tuning. The second model, termed Ensemble, is based on ensemble learning and is a feature-based approach. Both models adopt the sequence-to-point (s2p) methodology [79], which will be explained in the following.

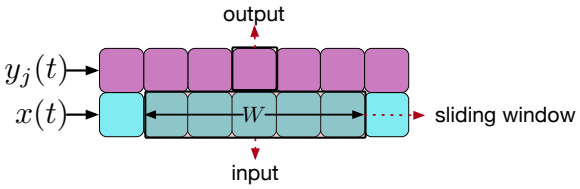


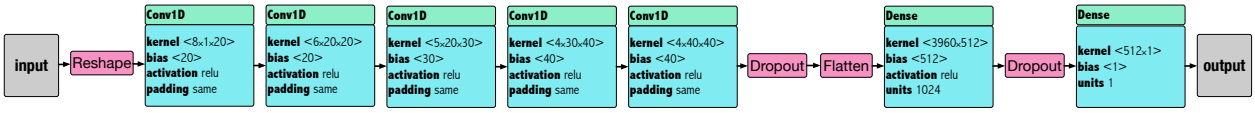
Figure 3.4: One training sample instance consisting of the aggregated power consumption and appliance j's power consumption data. The sliding window size is $W = 5$ in this example.

3.4.1 Sequence-to-point Method

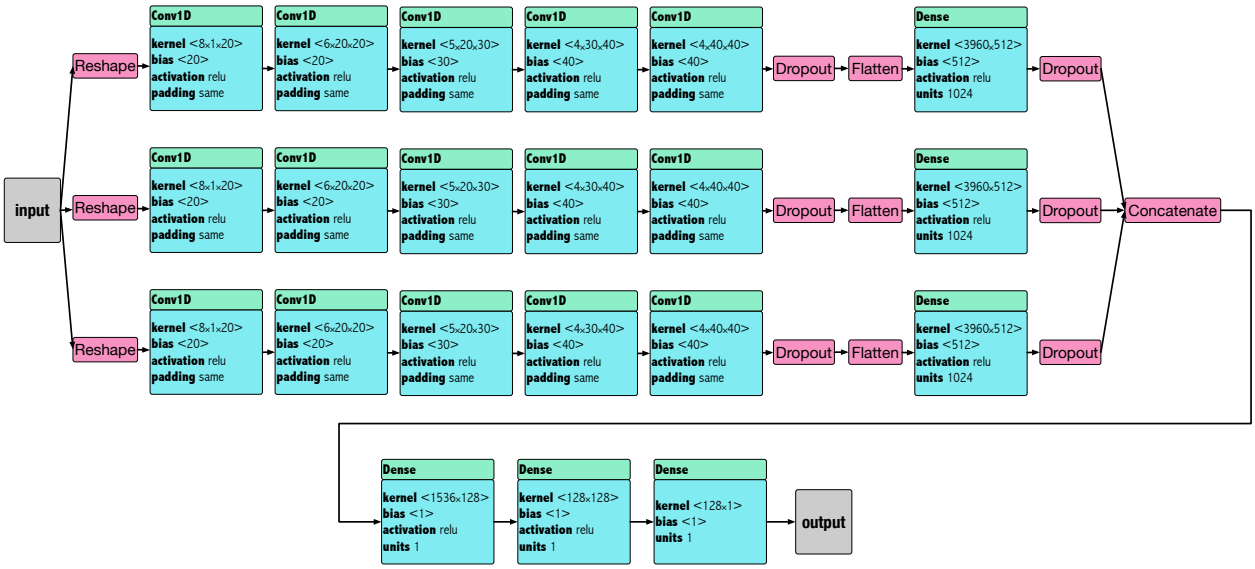
Traditional NILM solutions are sequence-to-sequence (seq2seq) learning, where a machine learning model maps an input sequence (i.e., the aggregate power consumption time series) to an output sequence (i.e., the power consumption time series of the target appliance). This method does not work well for NILM, where the extremely long sequences requires more memory and may cause the vanishing gradient problem in the training process. Although using a sliding window of size W could help to address the limitations, each $y_i(t)$ will be predicted W times, leading to larger errors at the edge [79].

Both our proposed methods utilize the s2p methodology instead [79]. S2p is motivated by the observation that an appliance's state at the center of the window is related to the aggregated power consumption samples before and after that point [1]. Therefore, a better prediction can be obtained for the center of the window using a full window of input data. In the example shown in Fig. 3.4, one training sample instance's input consists of the aggregate power consumption samples in a sliding window of size W. The learning model uses this window of input to predict the appliance's consumption at the midpoint of the window. In [92], the authors found that the s2p model outperformed 11 other power consumption disaggregation algorithms.

The s2p architecture used in this chapter is shown in Fig. 3.5(a), which consists of five convolutional layers followed by several dense layers. We also incorporate the dropout technique to deal with the overfitting problem [93]. Mean-square Error (MSE) is used as the default loss function for training the model.



(a) The s2p architecture used in the MAML based model.



(b) The architecture of the ensemble learning feature based pre-training model.

Figure 3.5: The architecture of the proposed pre-training neural network models.

3.4.2 MAML-based Approach

The first proposed solution is a fine-tuning method that is based on Model-Agnostic Meta-Learning (MAML), which is illustrated in the upper part of Fig. 3.6. First, the pre-training set is sampled to obtain meta-learning's training and validation sets, which are used to pre-train the base learner (i.e., the s2p model given in Fig. 3.5(a)). When applied to a new dataset, a small new fine-tuning set will be used to fine-tune the pre-trained model to achieve good transferability.

Gradient-based meta-learning is regarded as an effective approach for few-shot learning. MAML is most widely used to adapt pre-trained models to new tasks by only using a few samples. It aims to find a good initialization of model parameters suitable for varying tasks (i.e., different datasets). For few-shot learning problems, only a small amount of data is fed into a pre-trained model for several gradient updates in the fine-tuning phase. In meta-training, MAML introduces two loops of training (i.e., the inner and outer loops). In the inner-loop, a base learner is trained

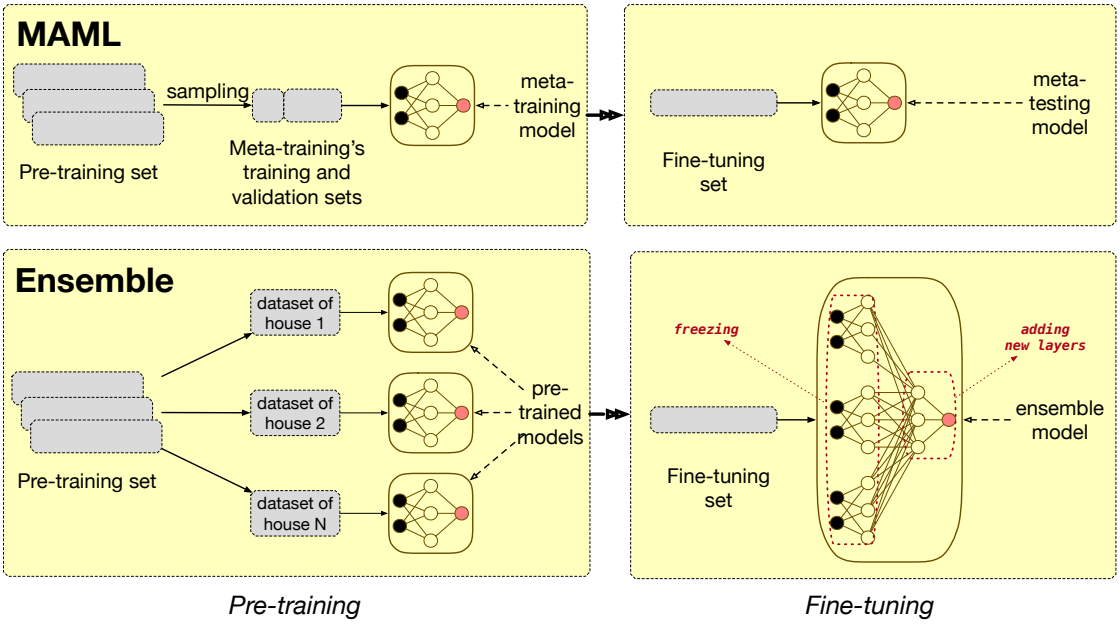


Figure 3.6: The proposed methods: (i) Upper: the MAML-based approach; (ii) Lower: the ensemble learning-based approach.

with $\mathcal{S}_i^{tr}$ by a base learning algorithm. In the outer-loop, a meta-algorithm updates the base learning algorithm to improve the model learned by the inner loop when dealing with new data $\mathcal{S}_i^{val}$, indicating the generalization performance of a model [94]. This is shown in the graphical model in Fig. 3.3 for task i. In the pre-training stage (i.e., meta-training), the base learner's parameters $\boldsymbol{\psi}$ are first initialized by $\boldsymbol{\theta}$, which is trained with the training set $\mathcal{S}_i^{tr}$ (in the inner-loop). The validation set $\mathcal{S}_i^{val}$ will be used to update the meta learner's parameters $\boldsymbol{\theta}$ (in the outer-loop). In the meta-testing stage, the pre-trained model is updated with additional gradient update steps using new data $\mathcal{T}^{tr}$ (i.e., fine-tuning). Instead of freezing the parameters of some layers, all the parameters $\boldsymbol{\theta}$ of the pre-trained model will be updated in the fine-tuning procedure. Finally, the well-trained model will be used for inference with new data $\mathcal{T}^{ts}$.

Formally, the problem that MAML solves in the meat-training stage is defined as follows

$$\min_{\boldsymbol{\theta}} \sum_{\text{Task } i} \mathcal{L}\left(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}_i^{\text{tr}}\right), \mathcal{S}_i^{\text{ts}}\right), \tag{3.8}$$

where $\boldsymbol{\theta}$ are the initialized parameters of the meta-leaner and base-leaner. The loss function of the inner-loop is defined as

$$\mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}_i^{tr}\right) = \mathbb{E}\left[\sum_{\mathbf{x},\mathbf{y}\sim\mathcal{S}_i^{tr}} \|f_{\boldsymbol{\theta}}(\mathbf{x}) - \mathbf{y}\|_2^2\right], \tag{3.9}$$

where $f_{\boldsymbol{\theta}}(\cdot)$ represents the inference model. In the inner-loop of the meta-training procedure, the base-learner's parameters are updated by

$$\boldsymbol{\psi}_i = \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}\left(\boldsymbol{\theta}, \mathcal{S}_i^{\mathrm{tr}}\right), \tag{3.10}$$

where α is the inner-loop's learning rate. In the outer-loop, the loss function is defined as

$$\mathcal{L}\left(\boldsymbol{\psi}_i, \mathcal{S}_i^{val}\right) = \mathbb{E}\left[\sum_{\mathbf{x},\mathbf{y}\sim\mathcal{S}_i^{tval}} \|f_{\boldsymbol{\psi}_i}(\mathbf{x}) - \mathbf{y}\|_2^2\right]. \tag{3.11}$$

MAML solves problem (3.8) by using stochastic gradient descent (SGD), which involves a gradient through a gradient (i.e., need to compute the Hessian matrix). To speed-up the training process, we do not calculate the Hessian matrix, but use its first-order approximation (i.e., the Jacobian matrix). The simplified MAML algorithm is presented in Algorithm 3.

3.4.3 Ensemble Learning based Approach

Our feature-based approach is motivated by ensemble learning [28], which aims to tackle the challenge of generalization i.e., to boost the performance of the pre-trained model on any unknown dataset. Ensemble methods, i.e., stacking, have been shown to be effective for time series forecasting problems [20]. Usually, data is partitioned by a clustering algorithm, and each cluster is used to train a first-level learner. Then another neural network is used as a second-level learner to fuse the outcomes from the first-level learners for improved forecasting results.

Algorithm 3: First-order MAML [91]

Require α: inner-loop step size;
Require β: outer-loop step size;
Require A: inner-loop epochs;
Require B: outer-loop epochs;
Require Dataset $\mathcal{S}$;
for $a = 1 : B$ **do**
    Sample the source set $\mathcal{S}$ for N times to obtain $\left\{\mathcal{S}_i^{tr}, \mathcal{S}_i^{val}\right\}_{i=1}^{N}$;
    for $i = 1 : N$ **do**
        for $a = 1 : A$ **do**
            Evaluate $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{S}_i^{tr})$;
            Implement gradient descent: $\boldsymbol{\psi}_i = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{S}_i^{tr})$;
        end
        Calculate gradient: $\nabla_{\boldsymbol{\psi}} \mathcal{L}(\boldsymbol{\psi}_i, \mathcal{S}_i^{val})$;
    end
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{i=1}^{N} \nabla_{\boldsymbol{\psi}} \mathcal{L}(\boldsymbol{\psi}_i, \mathcal{S}_i^{val})$;
end

The architecture of the proposed ensemble learning based model, termed Ensemble, is illustrated in the lower part of Fig. 3.6. Since the data from each house naturally form a cluster, the clustering algorithm is not needed here. The data from each house is used to train a first-level learner (i.e., a pre-trained model). As in the MAML based approach, a similar architecture of five convolutional layers followed by dense layers is adopted for the pre-trained models, as shown in Fig. 3.5(b). Similarly, dropout is incorporated to mitigate overfitting [93]. The ensemble model then integrates the outcomes (except for the last layer) from the pre-trained learners with a concatenate module followed by several dense layers to provide the final prediction. The fusion process in fine-tuning is to select a proper combination of the feature extractors (pre-trained learners) to deal with unknown data. Due to the diversity of the feature extractors, each for a suitable case, as well as a well-designed fusion model, the ensemble model is suitable for adapting the pre-trained models to unknown datasets.

As shown in the lower part of Fig. 3.6, our ensemble model has two phases of training, i.e., pre-training and fine-tuning. In the pre-training phase, we split the original pre-training set $\mathcal{S}$ into several subsets, each consisting of the data from a different house. Each subset will be used to

train an s2p model. In the fine-tuning phase, we first freeze each base learner's parameters. Then we concatenate all the parameters from every base model except the last dense layer as feature extractors. Three layers of a fully-connected deep neural network is then used to combine these feature extractors (i.e., pre-trained learners), as shown in Fig. 3.5(b). The parameters of the dense layers are trained with the fine-tuning set $\mathcal{T}^{tr}$.

3.5 Dataset and Experiment Setup

We evaluate the performance of the two proposed methods with extensive experiments using open-source NILM datasets. They are compared with several baseline schemes, e.g., traditional transfer learning, to validate their advantages. The datasets used in the evaluation and the experiment configurations are presented in this section.

3.5.1 Datasets

We use two real-world datasets, REFIT [4] and UK-DALE [5], to evaluate the performance of the proposed energy disaggregation methods. Both datasets are from England and provide house-level aggregate energy consumption and individual appliances' power consumption data measured by sensors deployed in the houses, while the households were conducting their usual domestic activities when the data was collected. The features of the two datasets are summarized in Tables 3.1 and 3.2, respectively.

In particular, the REFIT dataset consists of data from 21 houses, while the UK-DALE dataset has data from five houses. The data in the REFIT dataset was recorded every 8 seconds, to mimic the data collected by the SMETS2 smart meter standard2 [4]. Each house was equipped with nine appliance monitors and one current transformer sensor. The time duration of the REFIT dataset was from September 2013 to July 2015. We use a cleansed version of the REFIT dataset, where the missing values in each house (i.e., the NaN values) have been either zeroed or forward filled. In the UK-DALE dataset, each house's aggregated power consumption was recorded every 1 or

Table 3.1: Appliances and Houses in the REFIT Dataset

Meta-training dataset (pre-training): the REFIT dataset [4]			
	Training and validation dataset		
Appliances	Houses	Time period	Samples (M)
Kettle	9, 12, 20	2013-12-07 to 2015-07-08	17.20
Microwave	10, 12, 17, 19	2013-11-20 to 2015-06-30	29.80
Washing Machine	2, 7, 9, 16, 17	2013-09-17 to 2015-07-08	19.92
Dish Washer	7, 9, 13, 16	2013-09-26 to 2015-07-08	23.38
Fridge	2, 5, 9, 12	2013-09-17 to 2015-07-08	31.33

Table 3.2: Appliances and Houses in the UK-DALE Dataset

Meta-testing dataset: the UKDALE dataset [5]			
	Training (fine-tuning) dataset		
Appliances	House	Time period	Samples (M)
Kettle, Microwave, Fridge, Dish Washer, Washing Machine	2	2013-5-20 to 2013-5-29	0.108
	Testing dataset		
Appliances	House	Time period	Samples (M)
Kettle, Microwave, Fridge, Dish Washer, Washing Machine	2	2013-5-30 to 2013-10-10	1.592
	Validation dataset		
Appliances	House	Time period	Samples (M)
Kettle, Microwave, Fridge, Dish Washer, Washing Machine	1	2012-11-9 to 2012-11-18	0.102

6 seconds, and each individual appliance was measured every 6 seconds. The 6-second dataset is used in our experiment. It should be noticed that the UK-DALE dataset has been preprocessed; but we use the original dataset as it is. In order to be consistent with the data in REFIT, the UK-DALE data are down-sampled to 8 seconds.

We apply standard score normalization in data before all the models are trained and tested. The value of each appliance's mean and standard deviation can be found in [1]. In our experiments,

Table 3.3: Hyper-parameters of the Proposed Models

MAML	
Window size	99
Batch size	2000
SGD (inner-loop step size)	0.001
Adam (outer-loop step size)	0.001
Meta-training inner-loop epochs	1
Max meta-training outer-loop epochs	50
Max meta-testing's training epochs	10
Ensemble	
Window size	99
Batch size	2000
Adam	0.001
Maximum pre-training epochs	50
Maximum fine-tuning epochs	10

the REFIT dataset is used for pre-training, while the UK-DALE dataset is used for fine-tuning and testing, to test the models' generalization performance.

3.5.2 Hyper-parameters and Neural Network Training

Detailed information of the hyper-parameters of the proposed models are summarized in Table 3.3. All the models are implemented with TensorFlow 2.2.0 and trained on NVIDIA RTX 2070 Mobile with the Ubuntu 18.04 operating system. The window size W is set to be 99, 299, and 499. We find that the difference in performance between the different window sizes is small. So we select the smallest value W for computational efficiency. For the meta-learning model (i.e., MAML), the inner-loop of meta-training has a step size $\alpha = 0.001$ using the SGD optimizer [95]. We implement one gradient update in the inner-loop. The outer-loop is solved using the Adam optimizer [44], which is implemented with 50 gradient updates. During meta-testing, all layers of the trained model are fine-tuned with the Adam optimizer with ten gradient updates. For the Ensemble model, the Adam optimizer is used for all the pre-trained base models.

3.6 Experiment Results and Discussions

3.6.1 Experiment Methodology

In this section, we evaluate the two proposed approaches and compare them with several baseline power disaggregation algorithms. We choose five appliances in the experiments, including kettle, fridge, washing machine, dishwasher, and microwave. Each model for the appliances is trained individually, which means, for every appliance, a distinct dataset (con-training both meta-training and meta-testing sets) is constructed and used.

As mentioned in Section 3.3, to test the transferability of the models, two different datasets are used for pre-training and fine-tuning, respectively. In our experiments, we use REFIT as pre-training dataset. This is because REFIT is a relatively large dataset, which is expected to be able to equip the trained model with better generalization ability. UK-DALE is used as the testing dataset, where the house 2 data is used to fine-tune and test the model, and the house 1 data is used as the validation set. The detailed dataset split information is provided in Table 3.1 and Table 3.2.

Two stages of learning are conducted. Take the fridge's model as an example. For MAML, during the meta-training process, data of houses 2, 5, 9, and 12 in REFIT are first used to pre-train the model (as shown in Fig. 3.5(a)). The *zero-shot* results are obtained by directly applying the pre-trained model for inference for house 2 in UK-DALE. The *few-shot* results are obtained by using a few house 2 data in UK-DALE to fine-tune the model and then using the fine-tuned model for inference for house 2 in UK-DALE. For Ensemble (i.e., the feature-based pre-train method), data of houses 2, 5, 9, and 12 in REFIT are used to pre-train multiple base models (as shown in Fig. 3.5(b)). During the fine-tuning process, we will first examine each model's performance without any parameter updates using the meta-testing's test data to obtain the zero-shot results. The best pre-trained model, which scores the highest performance on house 2 in UK-DALE, will be fine-tuned with new data in the same way as MAML to obtain the few-short results.

The following three baseline schemes are used in our comparison study:

- Sequence-to-point (s2p): this is the model shown in Fig. 3.5(a) that is trained from scratch using only meta-testing's fine-tuning dataset (see Table 3.2). This is regarded as the bottom-line benchmark.

- Transfer learning for NILM (TL) [1]: this is the traditional transfer learning approach that uses s2p as the base model. It is trained with REFIT and tested on UK-DALE, with and without fine-tuning.

- Pre-trained sequence-to-point (pre-s2p) model uses the REFIT dataset to train the base model, which is similar to TL [1]. The difference between pre-s2p and TL [1] is that data from different houses is used to build several models in pre-s2p, while TL [1] utilized the entire dataset to build only one model. We only choose the base model with the best zero-shot MAE performance for fine-tuning.

Note that the authors in [1, 92] compared the s2p scheme with other traditional machine leaning methods, and found s2p achieved the best performance. Therefore, we choose s2p as a baseline scheme in this section.

Two performance metrics are used in the evaluations, which is the mean absolute error (MAE) and the signal aggregate error (SAE). These two metrics are defined as follows.

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_j(t) - y_j(t)| \tag{3.12}$$

$$\text{SAE} = \frac{1}{r_j}\left|\hat{r}_j - r_j\right|, \tag{3.13}$$

where $\hat{y}_j(t)$ and $y_j(t)$ are the estimated power consumption of appliance j and the ground truth, respectively; T is the duration of the time period; and $\hat{r}_j$ and r_j are the predicted total energy consumption and the ground truth of appliance j, respectively. MAE is used to measure the difference between the predict appliance power usage at every time instance and the ground truth of the appliance. SAE shows the relative error of the total energy consumption of the appliance [1].

3.6.2 Results and Discussions

Kettle	**Zero-shot**		**Few-shot**	
Model	MAE	SAE	MAE	SAE
s2p	-	-	21.287	0.367
TL [1]	6.260	0.060	16.879	0.043
pre-s2p model 1 (house 9)	6.124	0.155	7.518	0.140
pre-s2p model 2 (house 12)	9.539	0.248	-	-
pre-s2p model 3 (house 20)	32.889	0.816	-	-
MAML (proposed)	12.485	0.198	5.817	0.043
Ensemble (proposed)	-	-	**3.424**	**0.008**

Dish washer	**Zero-shot**		**Few-shot**	
Model	MAE	SAE	MAE	SAE
s2p	-	-	20.552	0.096
TL [1]	16.490	0.130	41.106	0.516
pre-s2p model 1 (house 5)	18.776	0.028	-	-
pre-s2p model 2 (house 7)	18.633	0.243	-	-
pre-s2p model 3 (house 9)	28.516	0.523	-	-
pre-s2p model 4 (house 13)	16.191	0.346	15.130	0.243
pre-s2p model 5 (house 16)	40.922	0.958	-	-
MAML (proposed)	17.882	0.361	**13.292**	0.254
Ensemble (proposed)	-	-	13.746	**0.033**

Washing machine	**Zero-shot**		**Few-shot**	
Model	MAE	SAE	MAE	SAE
s2p	-	-	9.574	0.733
TL [1]	14.840	0.500	22.941	0.899
pre-s2p model 1 (house 2)	9.629	0.679	-	-
pre-s2p model 2 (house 7)	8.356	0.626	7.751	0.431
pre-s2p model 3 (house 9)	9.631	0.785	-	-
pre-s2p model 4 (house 16)	11.070	0.767	-	-
pre-s2p model 5 (house 17)	8.613	0.600	-	-
MAML (proposed)	9.332	0.648	7.315	0.487
Ensemble (proposed)	-	-	**5.179**	**0.288**

Microwave	**Zero-shot**		**Few-shot**	
Model	MAE	SAE	MAE	SAE
s2p	-	-	13.174	1.194
TL [1]	4.770	0.080	10.973	0.019
pre-s2p model 1 (house 10)	4.767	0.345	4.498	0.259
pre-s2p model 2 (house 12)	7.739	0.755	-	-
pre-s2p model 3 (house 17)	6.849	0.112	-	-
pre-s2p model 4 (house 19)	5.275	0.093	-	-
MAML (proposed)	5.275	0.093	**3.215**	0.120
Ensemble (proposed)	-	-	3.490	**0.018**

Table 3.4: Performance When Transferred to UK-DALE

Fridge	**Zero-shot**		**Few-shot**	
Model	MAE	SAE	MAE	SAE
s2p	-	-	28.842	0.109
TL [1]	17.000	0.090	33.078	0.266
pre-s2p model 1 (house 2)	27.793	0.191	-	-
pre-s2p model 2 (house 5)	30.245	0.165	-	-
pre-s2p model 3 (house 9)	26.497	0.085	26.210	0.116
pre-s2p model 4 (house 12)	30.787	0.417	-	-
MAML (proposed)	27.714	0.280	**16.562**	**0.068**
Ensemble (proposed)	-	-	16.887	0.088

The evaluation results (i.e., MAE and SAE) are presented in Table 3.4, where zero-shot means the pre-trained models are tested on the testing set directly, and few-shot means the pre-trained models' parameters are updated with the fine-tuning set and then the fine-tuned models are tested on the testing set. There are no zero-shot results for s2p and Ensemble, since s2p is trained from scratch using the fine-tuning data and Ensemble requires fine-tuning data to combine the individual pre-s2p models. We use 10K fine-tuning sample instances' for updating the model parameters, which is collected on the first day of the fine-tuning set. The results of the transfer learning method (TL) proposed in [1] are presented as well. We found that with TL, the pre-trained model with fine-tuning performs even worse than the one without fine-tuning. We include both TL results with or without fine-tuning in the table. The parentheses following each pre-s2p model indicate the specific house in dataset REFIT used to pre-train the model. Only the pre-s2p model that achieves the best MAE performance for zero-shot of learning will be further updated with fine-tuning.

As can be seen from Table 3.4, both proposed pre-trained methods, i.e., MAML and Ensemble, outperform the traditional machine learning and transfer learning methods for all the tested appliances with respect to MAE and SAE. Next, we analyze the results in more detail in the following.

From Zero-shot to Few-shot

By updating the parameters (i.e., from zero-shot to few-shot fine-tuning), the transfer learning method used in [1] got an even worse result, with an -598.71% average improvement in MAE. This is because the TL method uses weak-relevant data in fine-tuning, which is the data from house 1 in UK-DALE. We try to diversify the data used for pre-training as in [1]. However, there is no guarantee that the data for fine-tuning comes from a similar distribution. Thus, we further improve fine-tuning by using only a small amount of data of house 2 (with no overlap with the unknown testing data). However, in some cases (e.g., pre-s2p model 1 of appliance kettle), negative transfer still happens, where the few-shot MAE (7.518) is slightly larger than the zero-shot MAE (6.124). Moreover, the improvements achieved by the pre-s2p models for other appliances are all insignificant. If we regard the pre-trained model's parameters as the neural network's starting point in the search space, the weight initialization of traditional transfer learning used for NILM is not optimal. Consequently, the DNNs get stuck in local minima with sub-optimal solutions.

The two proposed methods overcome this problem. On one hand, MAML achieves 53.41%, 25.67%, 21.61%, 40.24%, and 39.05% improvements in MAE for the kettle, dishwasher, washing machine, fridge, and microwave, respectively. On the other hand, Ensemble achieves 32.59%, 37.20%, 44.68%, 41.20% and 41.22% improvements in MAE compared to all the pre-trained models it uses.

With or Without Pre-training

We also compare the pre-trained models with the one trained from scratch (i.e., s2p). From the table, we can see that the best pre-trained model always outperforms s2p when using the same 10k

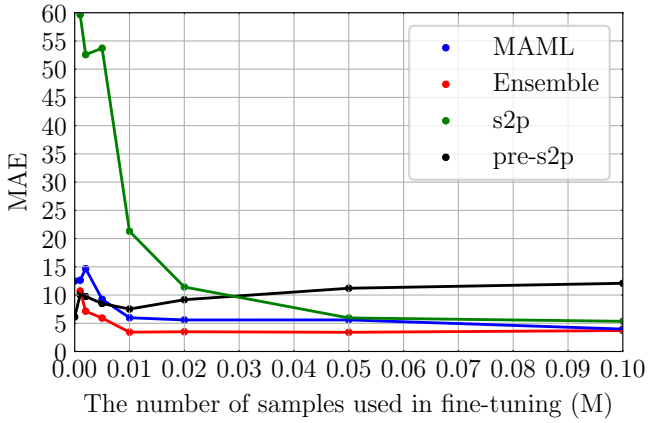


Figure 3.7: The relationship between models' performance (MAE) and amount of new samples used in additional 10 gradient updates for appliance kettle.

new data samples. The improvement in MAE are 83.92%, 35.23%, 45.91%, 42.58%, and 75.60%, respectively, for different appliances. The improvement in SAE are 83.92%, 35.23%, 45.91%, 42.58%, and 75.60%, respectively, for the appliances.

We next investigate how much fine-tuning data is needed to achieve a good performance on the NILM task. We further expand the results for appliance kettle in house 2 in UK-DALE with sample size increased from 0 to 100k. The new MAE results are shown in Fig. 3.7. The pre-s2p model is pre-trained with house 9 data in REFIT. As can be seen, except for transfer learning (pre-s2p), all other methods, including the model trained from scratch (s2p), achieve improved performance when more samples are used in fine-tuning. The pre-s2p model again suffers from the negative transfer problem, no matter how many samples are used to fine-tune its parameters. The MAEs of the two proposed methods (MAML and Ensemble) are initially (i.e., zero-shot) lower than that of s2p, and quickly reduces to stable values when 10K samples are used in fine-tuning. We also find the ensemble model outperforms MAML with a slightly smaller MAE. The s2p model needs at least 50k new samples to achieve the same MAE as MAML and at least 100K new samples to achieve the same MAE as Ensemble.

Fig. 3.8 presents an ablation study of validation error for appliance kettle using house 1's data in UK-DALE with different gradient steps for few-shot learning. We observe that all methods

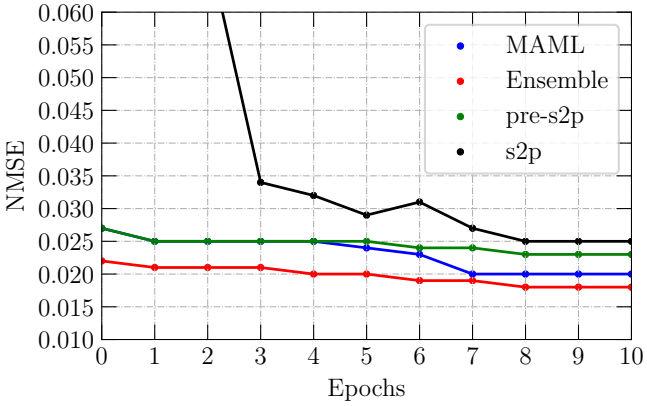


Figure 3.8: Validation error (NMSE) of appliance kettle using 10k samples for domain adaptation in fine-tuning.

continue to improve (with a decreasing Normalized Mean Square Error (NMSE)) as there are more gradient steps, and the NMSEs converge to stable values after 8 gradient updates. The NMSE of the model trained from scratch (i.e., s2p) drops dramatically and is the highest among the four schemes. The two proposed methods both achieve smaller errors than the transfer learning model (i.e., pre-s2p).

Feature-based vs. Fine-tuning-based

We also plot the predicted power consumption values along with the ground truth for the five appliances, as well as the aggregated power consumption in house 2 in UK-DALE, including kettle, dishwasher, microwave oven, washing machine, and fridge, obtained with the four methods for a specific time period in Fig. 3.9. For each appliance, we include a zoomed-in plot of the curves as well as a plot for the entire time period. Since the same legend is used in all the plots, we only show the legend in Fig. 3.9(b) to make the plots more readable. The aggregated consumption is in the shade of light gray and the ground truth of the target appliance is in the shade of dark gray. It can be seen that the s2p model fails to predict the appliance's power consumption at some time instances, i.e., the appliance's state is off but it is predicted as on. This is quite obvious in Fig. 3.9(e) from 200 to 250, and from 320 to 350. The transfer learning model (pre-s2p) tends to overestimate the appliance's power consumption (e.g., see Fig. 3.9(c)) or underestimate it (e.g., see

Table 3.5: Execution Time and Model Size of the Proposed Models for Kettle

Model	Model size (MB)	Training time (Min)	Fine-tuning time (Min)
MAML	16.3	953.72	0.68
Ensemble	49.0	1173.06	2.01

Fig. 3.9(g)). The two proposed methods' predictions match the ground truth much more closely than the other two schemes.

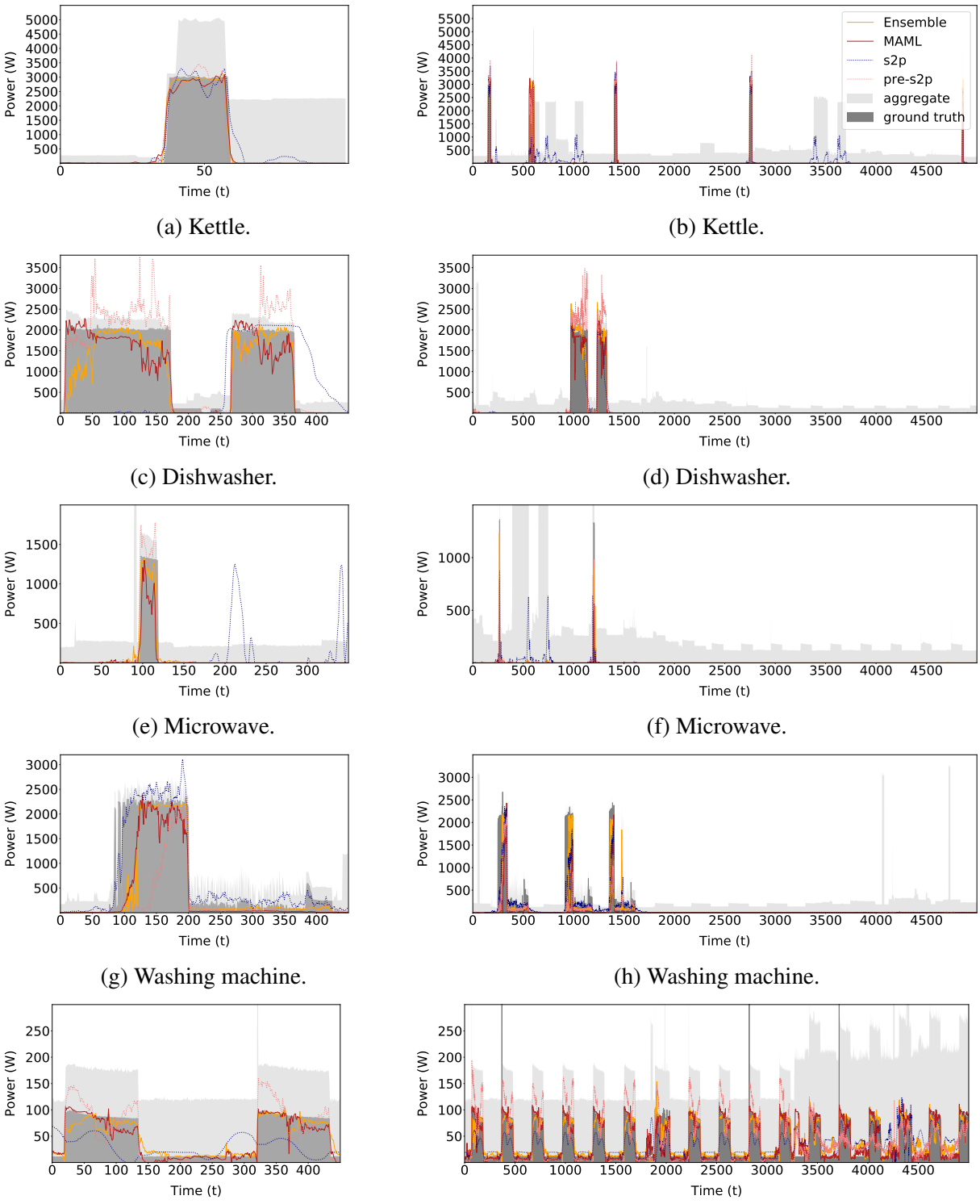
For the two proposed methods, it can be seen from Table 3.4 that they achieve similar MAE performance for dishwasher, fridge, and microwave. For kettle and washing machine, Ensemble outperforms MAML in MAE with an improvement ratio of 41.13% and 29.20%, respectively. Ensemble also outperforms MAML by achieving a smaller SAE for the kettle, dish washer, washing machine, and microwave. While both methods achieve good prediction performance, the Ensemble results are slightly better than that of MAML. This may due to the fact that MAML is a fine-tuning based approach; training all its parameters using a small amount of fine-tuning data may result in the overfitting problem [96].

Computational Complexity and Execution Time

In Table 3.5, we present the execution time and model size of the proposed models. The pre-training time of the Ensemble model given in the table is the accumulative time consumed by individual models. The results show that the Ensemble model requires more training time and larger model size than MAML. Training the base models in parallel on multiple GPUs will greatly improve the time efficiency of the pre-training process of the Ensemble model. Due to limited computing resources, we did not use this method in our experiments.

Limitation and Future Work

Due to limited datasets, the pre-training dataset and testing dataset are from the same country. The generalization of the pre-trained models across different countries needs to be further studied, as



(a) Kettle.

(b) Kettle.

(c) Dishwasher.

(d) Dishwasher.

(e) Microwave.

(f) Microwave.

(g) Washing machine.

(h) Washing machine.

(i) Fridge.

(j) Fridge.

Figure 3.9: Comparison of predicted appliance power consumption obtained by Ensemble, MAML, sequence-to-point (s2p), and transfer learning (pre-s2p) models with ground truth for five appliances (i.e., kettle, microwave, fridge, washing machine, and dishwasher) with the house 2 meta testing set.

differences in appliances and usage behavior between different regions may be larger. Furthermore, we only investigate the transferability of the pre-trained models predicting the same type of appliance between different regions. Transferability among diffffernt types of applicances would be an interesting problem to study. Finally, deep learning models are vulnerable to designed adversarial training samples. Attackers may fool the training process by introducing tainted data in the fine-tuning data. It is essential to improve the security and robustness of the pre-trained deep learning model.

3.7 Conclusions

In this chapter, we developed two types of pre-trained models based on CNNs for solving the NILM problem with a focus on generalization. The Ensemble model uses a neural network to connect several trained base models, and few-shot learning fine-tuning to adapt to a new task. The MAML approach initializes the pre-trained model with good weights, and can quickly adapt to a new task with a few gradient updates. The proposed pre-trained models can effectively solve the NILM problem. Compared to transfer learning, our models require fewer data for adaptation, and can quickly adapt to new NILM tasks. In addition, our proposed methods outperform transfer learning with respect to prediction accuracy and can effectively avoid negative transfer. The proposed schemes are validated with two open-source datasets and comparison with the baseline schemes.

Chapter 4

Middle Window Transformer for NILM

4.1 Introduction

The recent advances in the Internet of Things (IoT) allow the deployment of uniquely identifiable objects that are organized in an Internet-like structure to enable smart homes to monitor, control, and manage the house appliances [97]. The communication paths constructed by the IoT integrate smart meters, home appliances, and renewable energy, in a Home Energy Management System (HEMSs) [15, 98]. With more and more IoT-enabled technologies being developed and deployed, the HEMS system will become more sustainable, more resilient, and more energy efficient [31].

One important application of the IoT in HEMSs is load monitoring. The built-in sensors in appliances provide individual appliance's energy consumption information to the HEMS in real-time, which can be analyzed to optimize the energy usage and achieve energy savings. However, there are several practical issues that need to be addressed. First of all, electrical appliances typically last up to decades. As a result, a household typically include both old and new generations of appliances. The legacy appliances may not be equipped with smart sensors and their electricity consumption data is usually hard to measure. Second, the cost of installing sensors to legacy appliances could be high, including both the sensor and installation cost, as well as the power usage cost. Third, consumers are more and more concerned about their privacy; they may not be willing to share the information about their appliances' power consumption.

Non-Intrusive Load Monitoring (NILM), which is to identify individual appliance's electricity consumption from the given aggregated smart meter data, provides a useful solution to the above problems [59]. Recently, deep neural networks (DNNs), such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been shown effective to address the NILM problem. Since 2017, the Transformer [3] and its variants have dominated the field of natural language process (NLP), achieving superior performance for tasks such as language translation, text analytics, smart assistants, and so on. This is largely due to Transformer's capability of using the attention mechanism to capture the long-range dependency in sequential data. For computer vision (CV) tasks, the Vision Transformer (ViT) [99] has been shown to outperform the popular CNN model. In our recent work [100], a deep spatio-temporal attention approach was developed to forecast the temperature of stored grain using meteorological data. Such successes in NLP, CV, and other fields have attracted researchers to investigate Transformer's application to the NILM problem.

Although some recent preliminary studies have demonstrated the high potential of Transformer for NILM [101, 102], there are still many challenges remain to be addressed. First is the tradeoff between computational complexity and the ability to track long range dependency in energy consumption data, which usually contains rich daily, seasonal, and even annual patterns. The self-attention mechanism is the core of Transformer, which has a quadratic time complexity with regard to the input sequence length [103]. Low complexity models are thus desirable to allow longer input sequences. Second is the dependency on data. Like most Deep Learning (DL) models, Transformer requires a large amount of high quality labeled data for training, specifically, each individual appliance's power consumption data. The cost of data collection, e.g., submetering, could be high. In addition, many users are unwilling to share their appliance's information due to concern of privacy breach. Third is the generalization or transferability of the well trained Transformer model. The existing Transformer-based NILM methods are trained and tested on the same dataset, or assume the training and testing sets share similar data distribution. The transferability of the models have not been fully investigated, including testing across different appliances and/or

across different datasets. NILM models with strong transferability are useful to achieve accurate predictions for different, unseen houses, different models or brands of appliances, various aging degrees of electronic circuits, and different residents' daily habits and usage behavior [98].

In this chapter, we propose a **Mid**dle Window Trans**former** model, termed Midformer, for NILM, which incorporates several novel designs and follows the pre-training and fine-tuning paradigm to address the above problems. To deal with the computational complexity issue, Midformer is designed as a more efficient Transformer variant tailored to the characteristics of the NILM problem. Specifically, we utilize patch-wise attention in Midformer, which reduces the input length compared to point-wise attention used in existing models [101, 102]. We further apply the cyclically shifted window technique to increase the receptive filed. The drawback of patch-wise attention is that it ignores the connection across patches. In Midformer, we feed both cyclically shifted input and the original input into the attention layer to preserve the connection across patches. To reduce computation, we only calculate full attention using the middle range of the input, instead of using the entire input. This allows Midformer to focus on the middle range of the input and achieve a linear time complexity with respect to the input length (i.e., the window size).

To address the transferability issue and reduce the dependency on data, we follow the pre-training and fine-tuning paradigm. First, we pre-train multiple transformers (for different appliances) by using one dataset. Then we test the performance of the trained models on unseen data in the same dataset. Next, we examine the relationship among different appliances, i.e., could a model pre-trained using one appliance's data in a house be used to predict the power usage of another appliance in another house? The authors in [1] used the model learned from washing machine data to predict the power consumption of other appliances. In this chapter, we obtain the pre-trained model (including CNN, RNN, Transformer, and the proposed Midformer) for five appliances (including kettle, dishwasher, fridge, washing machine, and microwave). We then fine-tune and test the pre-trained model on a different dataset including the same and different appliances' data. With this approach, models do not need to be retrained from scratch for unknown houses and unseen

appliances, and can quickly adapt to new tasks with few-shot fine-tuning due to the well initialized parameters in the pre-trained models. This way, the computation in modeling training can be reduced and the dependency on data can be relieved.

We evaluate the performance of the proposed Midformer model using two real-world datasets and compare it with three baseline models including CNN, RNN, and Transformer. Our experimental study demonstrates the superior performance and great transferability of the proposed Midformer model for NILM problems over the state-of-the-art baseline models.

We organize the remainder of this chapter as fellows. We introduce related work in Section 4.2. In Section 4.3, we formulate the NILM problem and introduces the preliminaries of Transformers. In Section 4.4, we present the proposed transformer method Midformer. We present the datasets and experiment setup, and discuss the experimental study in Section 4.5. Finally, Section 4.6 concludes this chapter.

4.2 Related Work

4.2.1 Non-intrusive Load Monitoring Models

In the literature, many prior studies have developed approaches for solving the NILM problem, which can be mainly divided into two categories: (i) unsupervised learning, and (ii) supervised learning methods. In this section, we will briefly introduce the existing solutions for NILM; more detailed reviews of the different approaches applied to solving NILM can be found in [59,77,104].

Unsupervised Learning

Unsupervised learning has the unique strength of not requiring labeled data. The additive factorial hidden Markov model (AFHMM) is one of the most widely used unsupervised learning approaches for NILM [66–69], which converts time series data into Hidden-Markov Models and Bayesian models to infer the possible states of different appliances. Another method of unsupervised learning approach is the Graph Signal Processing (GSP) based method, which has also been

shown to be quite effective for NILM [71, 72]. The main drawbacks of these methods is that the prior domain knowledge needs to be provided, and such schemes may not perform well for solving problems that have a large number of appliances [1].

Supervised Learning

Supervised learning aims to learn a function, which maps an input to an output, from given input-output examples, i.e., labeled data. In the literature, various supervised learning methods have been applied to solving the NILM problem, such as Support Vector Machine [73], Decision Tress [74], K-Nearest Neighbours (k-NN) [76], and so forth. Recent works in this area demonstrate the promise of entirely deep learning approaches, such as Convolutional Neural Networks (CNNs) [79–81, 105], Long Short-Term Memory (LSTM) or its variant Gated Recurrent Units (GRUs) [2, 82, 106, 107], and denoising autoencoder [83, 108]. The main limitation of supervised learning (machine learning) method is that it requires large amounts of high quality training data. Such approaches usually require high computational power and storage capacity.

4.2.2 Transfer Learning for NILM

Most of the approaches applied to the NILM problem are carried out on the same data domain, which means the model is trained and tested using the same appliance's data in the same dataset. Very few previous studies have addressed the study of generalizability of the NILM models, also referred to as the transferability of the pre-trained models. For example, in [62], Murray et al. trained two different networks based on CNNs and RNNs, respectively, by using one of the three datasets and verify the models' transferability as well as generalization through the other datasets. However, the stability of the trained models is unsatisfactory due to the different data distributions in different databases, which lead to the poor domain adaptation performance.

To address this problem, D'Incecco, Squartini, and Zhong in [1] pre-trained their sequence-to-point (seq2point) learning model using the washing machine data in one specific dataset, and

then tested their pre-trained model on data of different appliances in different datasets. Fine-tuning, which is to train the pre-trained model using a small amount of examples from the testing dataset [98], was applied to adapt the pre-trained model to the difference between the different training and testing domains. However, the distribution of data used in fine-tuning was quite different from that of the tested house data, which led to the *negative transfer* effect. The generative adversarial networks (GANs) model has been applied to address the domain adaptation problem in NILM as well [88, 89], which was to train the feature generator and the domain discriminator in the adversarial manner. The limitation of this method is that training GANs requires finding a Nash equilibrium of a non-convex game with continuous high-dimensional parameters, which could fail to converge [109, 110]. Our previous work [98] developed a meta-learning based approach and an ensemble learning based approach that require fewer new data for adaptation, and can quickly adapt to new NILM tasks. However, we only explored the transferability between different datasets of same appliance in [98].

4.2.3 Transformer-based NILM Models

Motivated by the success of the Transformer architecture in many domains, most importantly in Natural Language Processing (NLP) [3], the self-attention[1] based Transformer has recently been proposed for NILM. The recent works [101, 102] both applied the attention mechanism to the feature maps extracted by CNNs to solve NILM tasks. The main drawback of these preliminary studies is that the computational complexity of self-attention grows quadratically with window (i.e., input) size, which could become a serious issue if the fixed window size is large. Moreover, the generalization performance of these models have not been verified through different datasets or appliances.

[1] "An attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence [3]."

4.3 Problem Statement

4.3.1 The NILM Problem

Consider a given collection of J time series $\{y_1(t),\ y_2(t),\ ...,\ y_J(t)\}_{t=1}^{T}$ that record the energy consumption of the J appliances in a house over a period of time T; $\{y_j(t)\}_{t=1}^{T}$ represents the power consumption of the jth appliance in the house. The aggregate power consumption $x(t)$ of the house at time t is calculated as follows.

$$x(t) = \sum_{j=1}^{J} y_j(t) + e(t), \tag{4.1}$$

where $e(t)$ is the measurement noise at time t. The NILM problem is to estimate the power consumption of an individual appliance from the given aggregate power consumption of the entire house. It is also called energy disaggregation since the goal is to separate the energy consumption measured at the aggregate level to that of individual appliances. It is non-intrusive since only the aggregate measurement is needed; and there is no need for submetering.

In NILM algorithms, to better handle the long time series data, usually a sliding/rolling window setting is adopted over the time series with a fixed window size, denoted by W, where the sliding/rolling step size is one. Rather than predicting a full window size of outputs, the NILM models often target at one single time instance (e.g., the middle point of the window) to avoid redundant computation. This approach is termed sequence-to-point (s2p) learning [79]. Therefore, given input data of total power consumption measurements over a window of size W, i.e., $\tilde{\mathbf{x}} = \{x(1), x(2), ..., x(W)\}$, the learning algorithm will compute output $\tilde{y}_j(\lceil W/2 \rceil)$, for all j.

4.3.2 Transformer and Multi-head Self-attention Mechanism

The Transformer model is based on the attention mechanism to significantly enhance the performance of deep learning, which computes the representation of a sequence by attending to information at different positions from different representation subspaces [3, 111]. The main idea of this

mechanism is to learn an alignment between each element in the sequence and others to decide which part of the sequence should be paid attention to [112].

For a given input sequence $\mathbf{I} \in \mathbb{R}^{W \times d_{model}}$, where d_{model} is the dimension of each data sample (i.e., length of the encoding vector), self-attention first transforms the input sequence into three matrices with three learnable weights. These three matrices are called queries, keys, and values, respectively, and they have the same depth of dimension d. Next the scaled dot-product is computed, which is given by:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \tag{4.2}$$

where $\mathbf{Q} = \mathbf{I}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{I}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{I}\mathbf{W}^V$, and $\mathbf{W}^Q \in \mathbb{R}^{d_{model} \times d}$, $\mathbf{W}^K \in \mathbb{R}^{d_{model} \times d}$, and $\mathbf{W}^V \in \mathbb{R}^{d_{model} \times d}$ are all trainable parameters that are used to map the input $\mathbf{I}$ into the three matrices $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$. The attention function (4.2) is similar to non-local means, which can be described as mapping a query and a set of key-value pairs to an output [3]. The weighted sum of the values is computed as the output of attention, where the weight is determined by the softmax score of the query with the corresponding key.

In the Transformer model, the attention processor is also called attention head. Multi-head Self-attention computes the self-attention score function describe in (4.2) on H different linear projections of queries, keys, and values in parallel. Then the results are concatenated as follows.

$$\text{MultiHead}(\mathbf{I}) = \text{Concat}\left(\sum_{i=1}^{H} \text{Attention}(\mathbf{Q_i}, \mathbf{K_i}, \mathbf{V_i})\right), \tag{4.3}$$

where $\mathbf{Q_i} = \mathbf{I}\mathbf{W}_\mathbf{i}^Q$, $\mathbf{K_i} = \mathbf{I}\mathbf{W}_\mathbf{i}^K$, and $\mathbf{V_i} = \mathbf{I}\mathbf{W}_\mathbf{i}^V$. The dimension of the learning parameters $\mathbf{W}_\mathbf{i}^Q$, $\mathbf{W}_\mathbf{i}^K$, and $\mathbf{W}_\mathbf{i}^V$ is $d_{model} \times d_i$, where $d_i = d/H$. By combining several similar attention results, the attention will have stronger power of discrimination.

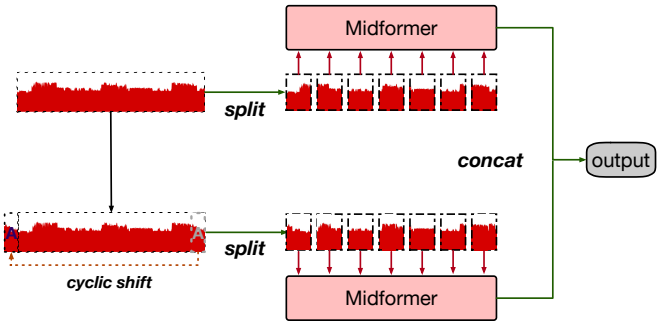
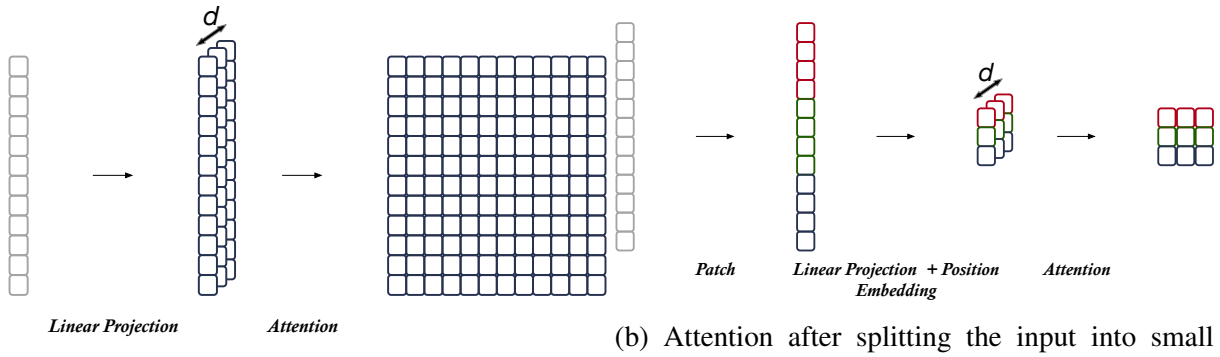


Figure 4.1: Architecture of the proposed Transformer-based approach, Midformer, to NILM.

4.4 Proposed Middle Window Transformer Approach

In this section, we introduce our proposed Transformer-based approach for NILM problems, which is termed middle window transformer (Midformer). Fig. 4.1 illustrates the overall architecture, which consists of four main parts, including (i) patch splitting and initializing, (ii) cyclic shift window, (iii) Transformer layers, and (iv) concatenation. Our intuition of designing this approach is to utilize the Transformer's attention ability to model the long range dependency in the energy consumption data, while reducing the computational cost.

The existing methods [101, 102] exploit the attention mechanism for NILM by combining CNNs with forms of self-attention. They first extract the feature map from input data by using convolutional layers. The extracted feature map is then fed into the Transformer layers. They both adopt global full self-attention in their models, which has a computational complexity that is quadratic to the size of feature map. For efficient modeling and computation, we leverage the technique proposed in the Vision Transformer (ViT) for image classification tasks [99], which reduces the context length by partitioning images into small patches and using the patches as input to the Transformer layers. A comparison of the existing approach and that adopted in this chapter is presented in Fig. 4.2. In particular, Fig. 4.2(a) shows the original point-wise attention projection method used in existing NILM works [101, 102], while Fig. 4.2(b) illustrates the patch-wise attention projection method adopted in this chapter. As shown in Fig. 4.2(a), when creating the attention matrix, the input will first be mapped into a space of depth d, which will then be used



(a) The global full self-attention.

(b) Attention after splitting the input into small patches.

Figure 4.2: Comparing the point-wise and the patch-wise attention pattern.

by the attention mechanism to calculate the attention matrix. The length and width of the attention matrix are the same as the depth of the space. From the comparison figure, we can visually see that the patch-wise attention incurs significantly less computation than the point-wise attention as the dimension of the attention matrix becomes much smaller.

4.4.1 Patch Splitting and Initialization

In the proposed Midformer model, the input $\mathbf{I} \in \mathbb{R}^{W \times d_{model}}$ is first split into a sequence of non-overlapping patches of fixed-size $\{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_k\}$, where $\mathbf{I}_i \in \mathbb{R}^{W/k \times d}$, for $1 \leq i \leq k$, and k is the number of patches. Each patch contains W/k samples, and is fed into a neuron network to be projected into a d-dimension vector. Different from [101, 102], before the input data is passed into the Transformer blocks, we do not need the convolutional layers to extract the feature map and increase the hidden size of the input sequence. This part of essential operation is replaced by individual neuron networks that project the patches. We also add position embedding to the projection to maintain position information in the data. The output of this projection is referred as patch embeddings.

4.4.2 Window Shifting

The patch-wise self-attention splits the input series of samples into non-overlap patches. However, this approach breaks the data correlation at patch boundaries and ignores the connection across

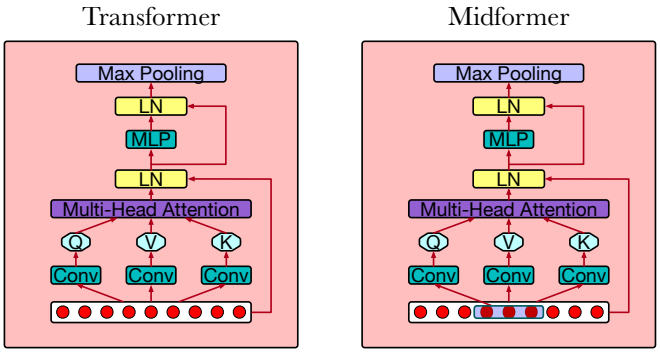


Figure 4.3: Comparison of Transformer and the proposed Midformer approach.

patches, which limit its modeling power. In order to capture the connection across patches while still maintaining the computational efficiency of non-overlapping patches, we apply the shifted window technique to broaden the receptive field, which is inspired by [113]. As illustrated in Fig. 4.1, we cyclically shift the input $\mathbf{I} \in \mathbb{R}^{W \times d_{model}}$ to the right for $W/(2k)$ positions (i.e., half of the patch size); the right-most half-patch of samples are moved to the left-most part of the window. The patches obtained from the cyclically shifted window of data are also fed into patch embeddings as well, to create a patch-wise feature map as shown in Fig. 4.1.

4.4.3 Transformer Layers

The two feature maps created by patch embeddings are then fed into the Transformer layers. We equally split the H heads into two parallel groups, where each group has $H/2$ heads (assume that H is an even number). One group accepts the feature map created from the original input, and the other group accepts the feature map created from the shifted input.

We follow the Transformer layer designed in [3], which consists of a multi-head attention layer and multi-layer perception (MLP) layer. A LayerNorm (LN) layer is applied after each attention layer and the MLP layer. A residual connection is used from the input to the first LN layer, and from the first LN layer to the second LN layer. The architecture of the original Transformer model is shown in the left plot in Fig. 4.3.

We further enhance the existing Transformer layer and propose the **Mid**dle Window Trans**former** (Midformer) layer, to achieve reduced computation complexity. The idea is simple: we only apply global attention on N (e.g., $N = 3$) input patches in the middle range of the window as queries, which is illustrated in the right plot in Fig. 4.3. The reason why we reduce the the number of queries is that, most NILM models (e.g., s2p [1]) only predict the appliance's power usage at the center position of the window. The middle range area of the window is where we should focus on. By reducing the number of queries, the complexity of the attention mechanism can be greatly reduced. It is worth noting that only the number of queries is reduced here, and the number of key-value pairs remains unchanged, which is fundamentally different from simply using a smaller window size W and then calculating the full attention. This technique contributes to the class of position-based sparse attention schemes, which reduce the required computations by limiting the number of query-key pairs that each query attends to [114].

4.4.4 Concatenation

Finally, an MLP (i.e., a fully connected layer) is utilized to concatenate the outputs of the two groups of Transformer blocks. The final MLP would restore the concatenated feature maps to the desired output size, which is one for NILM problems.

4.4.5 Computational Complexity Analysis

Supposing each input's dimension is $W \times d_{model}$, the patch size is W/k, the learnable parameter's dimension is $d_{model} \times d$, and the number of queries used in the Midformer layer is N. The computational complexity of the global Multi-head Self-attention module in each layer is $\mathcal{O}\left(W^2 \cdot d\right)$. The high cost of computing the global limits its ability to handle the usually large window sizes in NILM problems. However, with the proposed Midformer model, the computational complexity of the Multi-head Self-attention module is reduced to $\mathcal{O}\left(N/k \cdot W \cdot d\right)$. In the Midformer design, both N and k are set proportional to the window size W (e.g., $k = W/9$ and $N = k/3 = W/27$

in our experiments). Therefore, the computational complexity of Midformer is now linear to the window size W.

4.5 Experimental Study

In this section, we introduce the datasets and the system configuration used in our experiments to evaluate the performance of the proposed Transformer model. We then present our experimental study of the proposed model and compare it with three baseline models.

4.5.1 Datasets

We use two real-world datasets, the REFIT dataset [4] and the UKDALE dataset [5] to evaluate the performance of the proposed energy disaggregation method. The REFIT and UK-DALE datasets are both recorded in England. They both provide house-level aggregate energy consumption as well as individual appliances' power consumption data. In particular, the REFIT dataset consists of data from 20 households. Both the aggregate and appliance levels' data were recorded every 8 seconds from September 2013 to July 2015. The UKDALE dataset includes data from five houses. Each house's aggregated energy consumption was recorded every 1 or 6 seconds, and the appliance level data was measured every 6 seconds. In order to be consistent with data in different datasets, the aggregate level and appliance level data are down-sampled to 8 seconds. Standard score normalization is applied in data preprocessing; each sample x in the dataset is normalized as $\hat{x} = (x - \bar{x})/S$, where $\bar{x}$ is the sample mean and S is the sample standard deviation. We follow the approach in [1] to set the sample mean and sample standard deviation values for each appliance.

Following the approach in our recent work [98], for pre-training, we use a large-scale NILM dataset: i.e., the REFIT dataset. Specifically, we use the data from three houses as the pre-training set and the data from two other houses as the testing set for each appliance. The specific houses used and the amount of data from REFIT used to pre-train the model are summarized in Table 4.1. We then use the UKDALE dataset to evaluate the generalization of the models. We use only a

Table 4.1: Appliances and Houses Used in the REFIT Dataset [4]

	Training and validation dataset		
Appliances	House	Time period	Samples (M)
Kettle	5, 7, 13	2013-09-26 to 2015-07-08	18.91
Dishwasher	4, 10, 12	2014-03-07 to 2015-07-08	19.36
Fridge	2, 5, 12	2013-09-17 to 2015-07-08	13.28
Washing Machine	5, 7, 18	2013-09-17 to 2015-07-08	19.80
Microwave	5, 7, 18	2013-09-26 to 2015-07-08	19.80
	Testing dataset		
Appliances	House	Time period	Samples (M)
Kettle	9	2013-12-17 to 2015-07-08	6.17
	20	2014-03-20 to 2015-06-23	5.17
Dishwasher	9	2013-12-17 to 2015-07-08	6.17
	16	2014-03-10 to 2015-07-08	5.72
Fridge	9	2013-12-17 to 2015-07-08	6.17
	15	2013-12-17 to 2015-07-08	6.22
Washing Machine	15	2013-12-17 to 2015-07-08	6.22
	17	2014-03-06 to 2015-06-19	5.43
Microwave	17	2014-03-06 to 2015-06-19	5.43
	19	2014-03-06 to 2015-06-20	5.62

small part of the data in House 2 of the UKDALE dataset to fine-tune the pre-trained model and the rest of the unseen data of House 2 to test the performance of the fine-tuned pre-trained model. There is no overlap between the testing data and the fine-tuning data. The detailed information of the house and data from the UKDALE dataset used in our experiment is summarized in Table 4.2.

4.5.2 Model and Experimental Setup

Next we introduce the experiment setup and the models used to address the NILM problem. The following three baseline models are evaluated for comparison purpose.

Table 4.2: Appliances and Houses Used in the UKDALE Dataset [5]

	Fine-tuning dataset		
Appliances	House	Time period	Samples (M)
Kettle, Dishwasher, Fridge, Washing Machine, Microwave	2	2013-5-20 to 2013-5-29	0.108
	Testing dataset		
Appliances	House	Time period	Samples (M)
Kettle, Dishwasher, Fridge, Washing Machine, Microwave	2	2013-5-30 to 2013-10-10	1.592

- Sequence-to-point (s2p [1]): this baseline model uses the same structure of sequence-to-point method as in [1].

- Bidirectional Gated Recurrent Units (Bi-GRU) [2]: this baseline model utilizes BiGRU, rather than LSTM, to reduce the amount of model parameters while maintaining a similar performance as the RNN model.

- Transformer (Transformer) [3]: this is the traditional Transformer model. It has the same hyper-parameters as the Midformer model proposed in this chapter, which are summarized in Table 4.3.

All the models are implemented using TensorFlow 2.6.0 and trained on NVIDIA RTX 2070 Mobile. We pre-trained all the models using the ADAM optimization algorithm [44] with a maximum of 50 gradient updates. We update the weights with a learning rate of 0.001 and use a mini-batch size of 100. Both Midformer and Transformer incorporate 2 to 4 attention layers. The projected dimension of Midformer is $d = 64$, and the number of heads is $H = 8$. The number of patches is fixed at $k = 9$. Table 4.3 describes the detailed information of the hyper-parameters.

We fine-tune the pre-trained model using the stochastic gradient descent (SGD) method with a momentum of 0.9 and a learning rate of 0.01.

Table 4.3: Hyper-parameter Setting of Midformer

Hyper-parameter	Value
Window size	99 297 495 693
Batch size	100
Adam	0.001
Maximum pre-training epochs	50
Maximum fine-tuning epochs	10
Number of heads	8
Number of layers	2 to 4
Patch size	9
Projected dimension	64

4.5.3 Performance Metrics

We use two metrics to evaluate the performance of the proposed Transformer model, which are the mean absolute error (MAE) and the signal aggregate error (SAE). These two metrics are defined as follows.

$$\text{MAE} = \frac{1}{T}\sum_{t=1}^{T}|\hat{y}_j(t) - y_j(t)| \tag{4.4}$$

$$\text{SAE} = \frac{1}{r_j}\left|\hat{r}_j - r_j\right|, \tag{4.5}$$

where T is the duration of the period used to predict the output; $y_j(t)$ is the ground truth of power consumption of appliance j and $\hat{y}_j$ is the predicted value by the NILM models; $\hat{r}_j$ and r_j are the predicted total energy consumption and the ground truth of appliance j over the period T, respectively.

4.5.4 Experimental Results and Discussions

Three scenarios are designed and examined in our experimental study, which are:

(i) The pre-trained model is evaluated on the same appliance in the same dataset;

(ii) The model is applied to a different dataset but on the same appliance;

(iii) The model learned using one appliance in one dataset is evaluated on other appliances in a different dataset.

Multiple cases are examined, which belong to these three scenarios and use the data from the two public datasets.

The REFIT Dataset

The results in terms of the evaluation metrics on the REFIT dataset are represented in Table 4.4, which covers Scenario (i) described above. In this experiment, models for each appliance is pre-trained using the REFIT training set. Next, the data for the same appliance from two unseen houses are used to evaluate the pre-trained model. For example, for kettle, the labeled kettle data from Houses 5, 7, and 13 are used to pre-train the models, and then the kettle data from houses 9 and 20 are used to test the pre-trained models, while all the houses belong to the same REFIT dataset. Table 4.4 shows that the proposed Midformer model achieves both lower MAE and SAE in most cases (i.e., 6 cases out of 10 for MAE and 7 cases out of 10 for SAE). The average MAE and SAE values are averaged over the two houses. Our model achieved the best MAE results in all the cases, as well as the best SAE results for all the cases except for fridge. Compared to the baseline model s2p [1], the MAE reductions for kettle, dishwasher, washing machine, microwave, and fridge are 35.21%, 18.23%, 16.54%, 9.35%, and 3.80%, respectively.

Fig. 4.4 presents the execution times of different models for training per epoch under different window sizes. The Transformer and Midformer models both have two attention layers. The training set includes 100K samples. From the figure, we can see that the s2p model [1] uses the least amount of time; our proposed model uses the second least amount of time. The traditional Transformer model, which has the same number of layers as Midformer, consumes the longest time for training. The Bi-GRU model [2] uses less training time than Transformer for most of the window sizes (except for $W = 100$K). However, it is still more time-consuming thank both s2p and Midformer.

Table 4.4: Model Performances on the REFIT Dataset

Kettle	**Testing House 9**		**Testing House 20**		**Average**	
Model	MAE	SAE	MAE	SAE	MAE	SAE
s2p	10.882	0.054	5.162	0.058	8.022	0.056
Bi-GRU	11.072	0.076	6.958	0.074	9.015	0.075
Transformer	7.874	**0.052**	**3.971**	0.056	5.923	0.054
Midformer	**6.313**	0.062	4.081	**0.041**	**5.197**	**0.052**

Dishwasher	**Testing House 9**		**Testing House 16**		**Average**	
Model	MAE	SAE	MAE	SAE	MAE	SAE
s2p	14.683	0.463	14.622	0.292	14.653	0.378
Bi-GRU	15.408	0.294	24.665	1.290	20.037	0.792
Transformer	14.159	0.463	21.533	1.348	17.846	0.906
Midformer	**12.973**	**0.154**	**10.991**	**0.289**	**11.982**	**0.222**

Fridge	**Testing House 9**		**Testing House 15**		**Average**	
Model	MAE	SAE	MAE	SAE	MAE	SAE
s2p	25.145	0.240	25.465	**0.095**	25.305	**0.168**
Bi-GRU	**23.672**	**0.107**	27.150	0.230	25.411	0.169
Transformer	29.153	0.366	**23.970**	0.507	26.562	0.437
Midformer	23.853	0.261	24.900	0.418	**24.377**	0.340

Washing Machine	**Testing House 15**		**Testing House 17**		**Average**	
Model	MAE	SAE	MAE	SAE	MAE	SAE
s2p	10.808	0.562	8.277	0.284	9.543	0.423
Bi-GRU	11.377	0.499	12.068	0.293	11.723	0.396
Transformer	10.573	0.562	10.373	0.244	10.473	0.403
Midformer	**9.236**	**0.281**	**6.694**	**0.229**	**7.965**	**0.255**

Microwave	**Testing House 17**		**Testing House 19**		**Average**	
Model	MAE	SAE	MAE	SAE	MAE	SAE
s2p	5.342	0.210	**3.984**	0.458	4.663	0.334
Bi-GRU	4.879	0.563	5.455	0.662	5.167	0.613
Transformer	6.505	0.594	4.507	0.266	5.506	0.430
Midformer	**4.395**	**0.190**	4.058	**0.250**	**4.227**	**0.220**

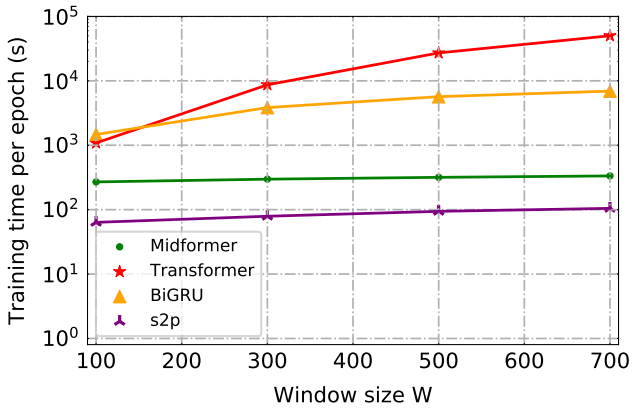


Figure 4.4: Execution times of s2p [1], Bi-GRU [2], Transformer (full attention) [3], and Midformer on the training set.

Considering the time and accuracy factors together, our proposed Midformer model consumes very little time for training and achieves the highest accuracy.

The UKDALE Dataset

In this experiment, we verify the transferability performance of the pre-trained model across different domains (i.e., different datasets and appliances). We first fine-tune the pre-trained model, which was originally learned using one appliance in the REFIT dataset, with a small portion of new data from the UKDALE dataset, and then use the test set of UKDALE to verify the performance of the model on the same or different appliance (see Table 4.2). These experiments cover Scerarios (ii) and (iii) described above.

The performance of the pre-trained models on the unseen UKDALE dataset is presented in Tables 4.5 and 4.6. Table 4.5 are the results of the pre-trained models without fine-tuning, while Table 4.6 are the results of the pre-trained models after fine-tuning, on the same appliance or an unseen appliance. The first column of the tables indicates the appliance and dataset learned by the pre-trained model. The second column indicates the unseen test dataset and corresponding appliances (same or different). The remaining columns are the MAEs and SAEs achieved by the four models.

Table 4.5: Results of the Pre-trained Model without Fine-tuning Tested on the UKDALE Dataset

Pre-trained Dataset	Testing Dataset	**Bi-GRU**		**s2p**		**Transformer**		**Midformer**	
REFIT	*UKDALE*	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE
Kettle	Kettle	16.579	0.163	22.642	0.442	16.204	0.268	15.099	0.234
	Dishwasher	72.909	0.498	66.627	0.669	70.220	0.562	71.449	0.558
	Fridge	47.079	0.783	44.102	0.786	46.458	0.798	46.625	0.795
	Washing machine	26.613	0.331	21.459	0.122	25.377	0.242	25.929	0.259
	Microwave	24.651	1.432	18.755	0.625	23.448	1.272	23.580	1.297
Dishwasher	Kettle	81.610	0.325	60.178	0.372	44.579	0.766	15.099	0.234
	Dishwasher	80.330	0.103	81.828	0.329	47.492	0.798	71.449	0.558
	Fridge	40.446	0.954	39.396	0.995	40.993	0.946	46.625	0.795
	Washing machine	44.862	0.601	41.812	0.939	44.541	0.632	45.929	0.259
	Microwave	44.487	0.294	44.362	0.362	42.864	0.263	43.580	1.297
Fridge	Kettle	371.049	8.047	318.502	6.556	306.362	6.264	306.362	6.264
	Dishwasher	379.803	3.338	305.632	4.484	308.999	4.471	313.630	4.569
	Fridge	25.034	0.321	25.621	0.873	24.273	0.218	25.702	0.222
	Washing machine	157.459	12.417	109.502	8.172	107.688	8.087	110.956	8.388
	Microwave	1224.569	28.545	169.378	20.964	168.284	20.870	172.590	21.404
Washing machine	Kettle	197.971	3.073	181.299	3.060	216.437	4.070	183.575	3.073
	Dishwasher	171.177	2.266	162.226	1.938	149.851	2.800	135.068	1.949
	Fridge	47.722	0.752	45.399	0.800	68.355	0.766	50.924	0.647
	Washing machine	31.097	1.337	19.867	0.432	58.687	3.966	33.079	1.359
	Microwave	79.898	8.825	66.123	7.040	99.997	11.766	65.980	7.091
Microwave	Kettle	113.396	1.038	119.416	1.233	56.051	0.424	116.196	1.096
	Dishwasher	121.416	0.515	129.421	0.670	62.974	0.624	129.026	0.627
	Fridge	39.345	1.000	41.747	0.919	39.344 1	.000	41.670	0.929
	Washing machine	11.629	0.995	17.279	0.471	11.726	0.985	16.329	0.545
	Microwave	9.974	0.657	6.429	0.167	9.271	0.718	7.698	0.202
average		98.444	3.144	86.360	2.537	83.619	2.943	84.566	2.586

From Table 4.5, we can see that the results of the pre-trained models without fine-tuning have relatively large errors. When the pre-trained model uses the same appliance as the test appliance, the test results are better than that using a different appliance. Except for the Bi-GRU [2] model, the other three models achieve similar MAE and SAE values, which are around 85 and 3, respectively.

From Table 4.6, it is obvious that fine-tuning has been very effective in reducing the error of all the models on unseen dataset and appliances, since both the MAEs and SAEs of all the models are greatly improved. For example, the average MAE of Midformer is reduced from 84.566 to 7.121, and the average SAE is reduced from 2.586 to 0.056, after fine-tuning (huge improvements). In the table, the bold numbers in each row indicate the *best result* among the four models obtained

Table 4.6: Results of Pre-trained Model with Fine-tuning Tested on the UKDALE Dataset

Pre-trained Dataset	Testing Dataset	**Bi-GRU**		**s2p**		**Transformer**		**Midformer**	
REFIT	*UKDALE*	MAE	SAE	MAE	SAE	MAE	SAE	MAE	SAE
Kettle	Kettle	10.064	0.116	9.854	0.106	10.585	0.318	**4.183**	**0.041**
	Dishwasher	39.001	0.409	**4.463**	**0.017**	14.511	0.153	5.135	0.040
	Fridge	34.475	0.025	24.296	**0.015**	36.476	0.139	**17.081**	0.061
	Washing machine	9.540	0.715	12.604	0.194	19.102	**0.126**	**6.979**	0.185
	Microwave	14.192	1.014	5.800	0.169	4.275	0.018	**4.183**	**0.013**
Dishwasher	Kettle	55.162	0.241	4.471	0.017	6.874	0.058	†**3.837**	**0.010**
	Dishwasher	29.180	0.389	**4.820**	**0.009**	7.970	0.086	6.974	0.012
	Fridge	34.949	0.030	28.004	0.196	36.237	0.113	**15.312**	**0.162**
	Washing machine	10.137	0.758	11.812	0.328	9.490	0.328	**5.301**	**0.049**
	Microwave	18.649	0.516	3.762	0.187	5.414	0.128	**3.177**	**0.065**
Fridge	Kettle	64.616	0.440	8.009	0.023	5.159	**0.018**	**4.520**	0.043
	Dishwasher	36.583	0.173	5.639	0.031	6.094	0.014	**5.251**	**0.006**
	Fridge	24.059	0.032	13.798	0.083	16.588	0.144	†**13.132**	**0.050**
	Washing machine	13.342	0.503	8.553	0.523	7.008	0.277	**5.110**	**0.045**
	Microwave	13.889	0.455	5.402	0.250	5.497	0.175	**3.126**	**0.171**
Washing machine	Kettle	37.098	0.435	5.994	0.027	7.482	0.014	**5.213**	**0.003**
	Dishwasher	30.852	0.266	†**4.254**	0.012	5.377	**0.003**	5.406	0.040
	Fridge	35.310	0.110	24.393	0.285	34.425	0.322	**14.992**	**0.032**
	Washing machine	15.987	0.177	8.089	0.390	9.523	**0.055**	†**4.887**	0.114
	Microwave	23.553	1.177	4.267	0.020	7.303	0.053	**3.165**	**0.041**
Microwave	Kettle	8.424	0.078	8.268	0.068	7.443	0.064	**6.114**	**0.020**
	Dishwasher	6.406	0.082	6.389	0.065	6.014	**0.029**	**4.310**	0.039
	Fridge	27.256	0.120	17.572	**0.012**	**17.428**	0.036	22.551	0.022
	Washing machine	14.767	0.661	13.904	0.380	6.806	0.374	**5.456**	**0.059**
	Microwave	5.456	0.126	5.289	0.090	4.541	0.039	†**2.630**	**0.028**
Average		24.518	0.358	10.088	0.136	12.145	0.123	**7.121**	**0.056**

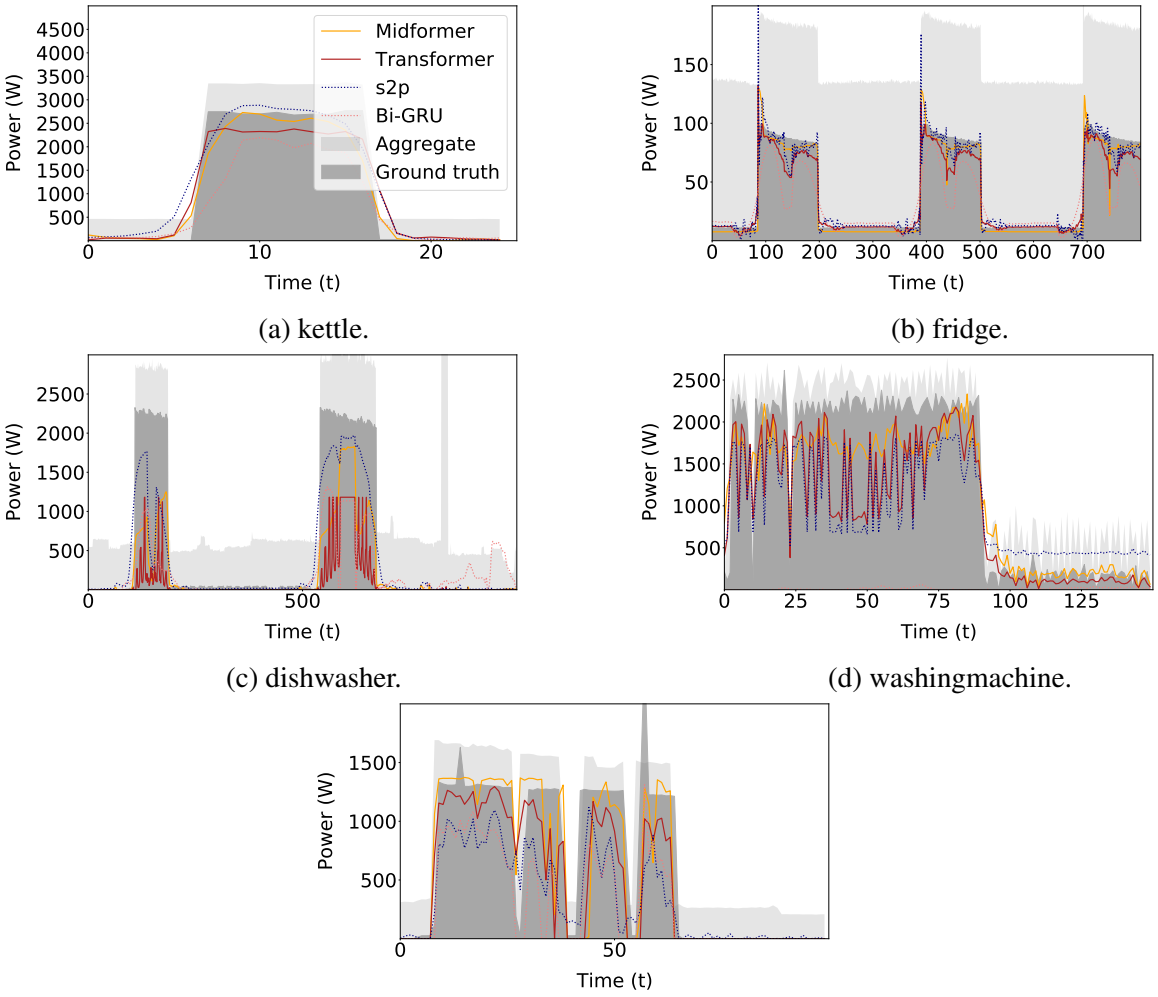
for the test set when using a pre-trained model of a particular appliance. For example, for pre-trained model using kettle in REFIT and the target appliance kettle in UKDALE, the Midformer model achieves the smallest MAE of 4.183 and the smallest SAE of 0.041. The number marked by symbol "†" indicates the *best model* for that target appliance among all the pre-trained models. For example, for the target appliance kettle, the pre-trained model of Midformer learned from the source appliance dishwasher schieves the best MAE of 3.837. To better present the results, we have summarized such information in Table 4.7.

We can make the following observations from these results. (i) The proposed Midformer model outperforms all other models on average and in most specific cases. (ii) Our proposed model achieves superior transferability performance, which means we can use the pre-trained Midformer model using one appliance for all other target appliances, resulting greatly reduced cost for model pre-training. (iii) In most cases, the best result for a target appliance is obtained with the model pre-trained using the same appliance. The best pre-trained model for fridge, washing machine, and microwave in the UKDALE dataset is the model learned from the same appliance in the REFIT dataset, respectively. This is intuitive since the pre-trained model will perform well if the test data and training data share similar features. In Table 4.7, the proposed Midformer model accounts for four of the five best results of transfer learning.

The predicted power consumption values of house 2 in the UKDALE dataset for the five appliances obtained by the four pre-trained models on the REFIT dataset (i.e., s2p, Bi-GRU, Transformer, and Midformer) for a specific time period are plotted in Fig. 4.5, along with the corresponding ground truth values. Note that the "Aggregate" values are the input to these models to be disaggregated into individual appliance's power consumption. The figure shows that the proposed Midformer model achieves the best performance compared to the three baseline models, except for dishwasher (which is consistent with the results in Table 4.7). The Bi-GRU model fails to predict the washing machine's power state at some time instances, i.e., the washing machine's state is on, but it is predicted as off (see Fig. 4.5(d)).

4.6 Conclusions

In this chapter, we proposed the Midformer model to tackle the NILM problem. We utilized patch-wise attention and reduced the query size to reduce the quadratic time complexity in traditional Transformer models to linear complexity. We also focused on the transferability performance of the models, which helped to reduce the model training cost and eased the deployment of the model in various environments. Our experimental study using two real-world datasets demonstrated the



(a) kettle.

(b) fridge.

(c) dishwasher.

(d) washingmachine.

(e) microwave.

Figure 4.5: Comparison of predicted power consumption values by Midformer, Transformer, s2p, and Bi-GRU for the five appliances, along with the ground truth values.

Table 4.7: Best Pre-trained Model for UKDALE Test Set

Test appliance in UKDALE	**Pre-trained appliance** in REFIT	**Model**	**MAE**	**SAE**
Kettle	Dishwasher	Midformer	3.837	0.010
Dishwasher	Washing machine	s2p	4.254	0.012
Fridge	Fridge	Midformer	13.132	0.050
Washing machine	Washing machine	Midformer	4.887	0.114
Microwave	Microwave	Midformer	2.630	0.028

superior performance and stronger transferability of the proposed Midformer model over three baseline, state-of-the-art models on addressing the NILM problem.

Chapter 5

Summary and Future Work

In this thesis, we discuss two deep learning applications among smart meter data. The first application is an LSTM based ensemble learning approach for short-term load forecasting. In the stacking framework, the first-level learner consisted of LSTM models, each trained with a data cluster; the second-level learner consisted of an FCC ensemble neural network for model fusion. An enhanced second-order optimization algorithm was proposed to solve the ensemble problem. The second application focuses on NILM problems. We first propose two novel pre- trained strategies based on ensemble learning and Meta-learning in NILM problems. These pre-trained models are more suitable for NILM problems than transfer learning models. We further explore the ability of the Transformers model to handle NILM problems. In order to solve the issue that Transformer models are expensive to compute, we proposed Middle Window Transformer, an efficient and effective Transformer-based approach for NILM tasks.

5.1 Future work

Potential future work can be categorized as follow.

5.1.1 Adversarial Attacks on DL-based NILM models

In Chapter 3 and Chapter 4, we proposed deep learning models to solve the NILM problem, which achieved very high accuracy. However, deep learning models are vulnerable to designed adversarial attacks, which can significantly degrade the accuracy of deep learning (DL) models for load

monitoring. The adversarial attack is an essential security concern for DNNs. It is interesting to study how to generate the adversarial samples and attack the DNN-based NILM identification schemes—improving the security and robustness of the trained NILM model. Future research could focus on the effects of non-targeted and targeted adversarial attacks on DNN-based NILM identification.

5.1.2 Multi-task Learning-Based Non-intrusive Load Monitoring

The original NILM was defined as a single-point blind source separation problem. Different appliances need to construct different training data sets. However, multiple appliances models use the same time series data as input. Thus, it can be transformed into multi-task problems, which provide multiple outputs for input in one run. The multi-task model for NILM is aware of the relationships among different appliances with less supervision and less computation. In light of the need for practical NILM, our future work could focus on designing multi-task learning algorithms that can accurately predict multiple appliances' power usage simultaneously [115].

5.1.3 Lifelong and Federated Learning for NILM

The proposed pre-trained models for NILM rely heavily on fine-tuning data. We performed the fine-tuned modeling in one go. However, the household system is evolving in many cases, e.g., replacing old and non-functioning appliances. Moreover, data security and user privacy have been a big concern by customers. The insufficient fine-tuning data limits pre-trained models' performance. Lifelong Learning integrates Federated learning could solve this problem. In this system, all data is stored locally, and the pre-trained model can be fine-tuned with the user's data and updated with the centralized system in real-time [116].

References

[1] M. D'Incecco, S. Squartini, and M. Zhong, "Transfer learning for non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 11, no. 2, pp. 1419–1429, Aug. 2019.

[2] O. Krystalakos, C. Nalmpantis, and D. Vrakas, "Sliding window approach for online energy disaggregation using artificial neural networks," in *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018, pp. 1–6.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, Long Beach, CA, Dec. 2017, pp. 5998–6008.

[4] D. Murray, L. Stankovic, and V. Stankovic, "An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study," *Scientific Data*, vol. 4, no. 1, pp. 1–12, Jan. 2017.

[5] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, vol. 2, no. 1, pp. 1–14, Mar. 2015.

[6] L. Wang, S. Mao, and B. Wilamowski, "Short-term load forecasting with LSTM based ensemble learning," in *Proc. IEEE GreenCom 2019*, Atlanta, GA, Aug. 2019, pp. 793–800.

[7] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, pp. 38:1–38:55, Sept. 2014.

[8] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Springer Mobile Networks and Applications Journal (MONET)*, vol. 19, no. 2, pp. 171–209, Apr. 2014.

[9] Asia Pacific Urban Energy Association, *Introduction to Sustainable Urban Energy Systems*, 2019. [Online]. Available: https://www.apuea.org/index.php/urban-energy-systems/introduction-to-urban-energy-systems2

[10] Y. Wang, S. Mao, and R. M. Nelms, "Online algorithm for optimal real-time energy distribution in smart grid," *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 1, pp. 10–21, July 2013.

[11] Y. Huang, S. Mao, and R. Nelms, "Smooth scheduling for electricity distribution in the smart grid," *IEEE Sysetms Journal*, vol. 9, no. 3, pp. 966–977, Sept. 2015.

[12] Y. Huang, S. Mao, and R. M. Nelms, "Adaptive electricity scheduling in microgrids," *IEEE Transactions on Smart Grid*, vol. 5, no. 1, pp. 270–281, Jan. 2014.

[13] ——, "Adaptive electricity scheduling in microgrids," in *Proc. IEEE INFOCOM 2013*, Turin, Italy, Apr. 2013, pp. 1142–1150.

[14] H. Zou, Y. Wang, S. Mao, F. Zhang, and X. Chen, "Distributed online energy management in inter-connected microgrids," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2738–2750, Apr. 2020.

[15] H. Zou, S. Mao, Y. Wang, F. Zhang, X. Chen, and L. Cheng, "A survey of energy management in interconnected multi-microgrids," *IEEE Access Journal*, vol. 7, no. 1, pp. 72 158–72 169, June 2019.

[16] Y. Wang, "Algorithms for optimal energy management in the smart grid," Ph.D. dissertation, 2015.

[17] A. Cooper, "Electric company smart meter deployments: foundation for a smart grid," *The Institute for Electric Innovation (IEI): Washington, DC, USA*, 2021.

[18] F. Uddin, "Energy-aware optimal data aggregation in smart grid wireless communication networks," *IEEE Trans. Green Commun. Netw.*, vol. 1, no. 3, pp. 358–371, Sept. 2017.

[19] M. Collotta and G. Pau, "An innovative approach for forecasting of energy requirements to improve a smart home management system based on ble," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 112–120, Mar. 2017.

[20] L. Wang, S. Mao, B. M. Wilamowski, and R. M. Nelms, "Ensemble learning for load forecasting," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 616–628, Apr. 2020.

[21] S. Li, L. Goel, and P. Wang, "An ensemble approach for short-term load forecasting by extreme learning machine," *Elsevier Appl. Energy*, vol. 170, pp. 22–29, May 2016.

[22] C. Cecati, J. Kolbusz, P. Różycki, P. Siano, and B. M. Wilamowski, "A novel RBF training algorithm for short-term electric load forecasting and comparative studies," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6519–6529, Oct. 2015.

[23] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *Proc. IEEE IECON 2016*, Florence, Italy, Oct. 2016, pp. 7046–7051.

[24] 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 Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[25] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, July 2018.

[26] Y. Wang, N. Zhang, Y. Tan, T. Hong, D. S. Kirschen, and C. Kang, "Combining probabilistic load forecasts," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3664–3674, July 2018.

[27] W. Zhang, H. Quan, and D. Srinivasan, "An improved quantile regression neural network for probabilistic load forecasting," *IEEE Transactions on Smart Grid*, vol. 10, pp. 4425–4434, July 2019.

[28] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL: Chapman and Hall/CRC, 2012.

[29] P. Smyth and D. Wolpert, "Linearly combining density estimators via stacking," *Springer Machine Learning*, vol. 36, no. 1/2, pp. 59–83, July 1999.

[30] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*. Burlington, MA: Morgan Kaufmann, 2011.

[31] Y. Wang, Y. Shen, S. Mao, X. Chen, and H. Zou, "Lasso & lstm integrated temporal model for short-term solar intensity forecasting," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2933–2944, Apr. 2019.

[32] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

[33] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM/SIAM SODA'07*, New Orleans, LA, Jan. 2007, pp. 1027–1035.

[34] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM Sigmod Record*, vol. 25, no. 2, pp. 103–114, June 1996.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD'96*, Portland, Oregon, Aug. 1996, pp. 226–231.

[36] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, pp. 205–206, 2017.

[37] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *Proc. 2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, Nov. 2017, pp. 33–42.

[38] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, p. Article No. 5, July 2015.

[39] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[40] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures–a comparative study," *IEEE Trans Ind. Informat.*, vol. 8, no. 2, pp. 228–240, May 2012.

[41] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE PIMRC 2017*, Montreal, Canada, Oct. 2017, pp. 1–6.

[42] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. ICML'10*, Haifa, Israel, June 2010, pp. 735–742.

[43] J. E. Vitela and J. Reifman, "Premature saturation in backpropagation networks: Mechanism and necessary conditions," *Elevier Neural Networks*, vol. 10, no. 4, pp. 721–735, June 1997.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint, arXiv:1412.6980*, Jan. 2017, [online] Available: https://arxiv.org/abs/1412.6980.

[45] J. Fan, "Accelerating the modified levenberg-marquardt method for nonlinear equations," *Mathematics of Computation*, vol. 83, no. 287, pp. 1173–1187, May 2014.

[46] J. Fan and J. Pan, "Convergence properties of a self-adaptive Levenberg-Marquardt algorithm under local error bound condition," *Springer Comput. Opt. Appl.*, vol. 34, no. 1, pp. 47–62, May 2006.

[47] ISO New England, "ISO New England Zonal Information," 2019. [Online]. Available: https://iso-ne.com/isoexpress/web/reports/pricing/-/tree/zone-info

[48] UC Irvine Machine Learning Repository, "Electricity Load Diagrams 2011-2014 Data Set," 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

[49] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Springer Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, Oct. 2003.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint, arXiv:1502.03167*, Mar. 2015, [online] Available: https://arxiv.org/abs/1502.03167.

[51] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, June 1997.

[52] Z. Niu, "Green communication and networking: A new horizon," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 3, pp. 629–630, Aug. 2020.

[53] M. Feng, S. Mao, and T. Jiang, "Boost: Base station on-off switching strategy for energy efficient massive mimo hetnets," in *Proc. IEEE INFOCOM 2016*, San Francisco, CA, Apr. 2016, pp. 1395–1403.

[54] ——, "Dynamic base station sleep control and RF chain activation for energy efficient millimeter wave cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9911–9921, Oct. 2018.

[55] ——, "BOOST: Base station on-off switching strategy for green massive MIMO HetNets," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7319–7332, Nov. 2017.

[56] S. Hu, X. Chen, W. Ni, X. Wang, and E. Hossain, "Modeling and analysis of energy harvesting and smart grid-powered wireless communication networks: A contemporary survey," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 461–496, Apr. 2020.

[57] G. W. Hart, "Nonintrusive appliance load monitoring," *Proc. IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec. 1992.

[58] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, "Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey," *MDPI Sensors*, vol. 12, no. 12, pp. 16 838–16 866, Dec. 2012.

[59] A. Ruano, A. Hernandez, J. Ureña, M. Ruano, and J. Garcia, "NILM techniques for intelligent home energy management and ambient assisted living: A review," *MDPI Energies*, vol. 12, no. 11, p. 2203, June 2019.

[60] C. Fischer, "Feedback on household electricity consumption: a tool for saving energy?" *Springer Energy Efficiency*, vol. 1, no. 1, pp. 79–104, May 2008.

[61] J. Leitão, P. Gil, B. Ribeiro, and A. Cardoso, "A survey on home energy management," *IEEE Access*, vol. 8, pp. 5699–5722, Jan. 2020.

[62] D. Murray, L. Stankovic, V. Stankovic, S. Lulic, and S. Sladojevic, "Transferability of neural network approaches for low-rate energy disaggregation," in *Proc. IEEE ICASSP 2019*, Brighton, UK, May 2019, pp. 8330–8334.

[63] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2012.

[64] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, May 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[65] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, July 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[66] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Proc. 2012 Int. Conf. Artif. Intell. Stat.*, La Palma, Canary Islands, Apr. 2012, pp. 1472–1482.

[67] M. Zhong, N. Goddard, and C. Sutton, "Signal aggregate constraints in additive factorial HMMs, with application to energy disaggregation," in *Proc. NIPS 2014*, Montreal, CA, Dec. 2014, pp. 3590–3598.

[68] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Proc. AAAI 2012*, Toronto, CA, July 2012, pp. 356–362.

[69] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *Proc. 2011 SIAM Int. Conf. Data Mining*, Mesa, USA, Apr. 2011, pp. 747–758.

[70] R. Bonfigli, E. Principi, M. Fagiani, M. Severini, S. Squartini, and F. Piazza, "Non-intrusive load monitoring by using active and reactive power in additive factorial hidden Markov models," *Elsevier Applied Energy*, vol. 208, pp. 1590–1607, Dec. 2017.

[71] B. Zhao, L. Stankovic, and V. Stankovic, "On a training-less solution for non-intrusive appliance load monitoring using graph signal processing," *IEEE Access*, vol. 4, pp. 1784–1799, Apr. 2016.

[72] K. He, L. Stankovic, J. Liao, and V. Stankovic, "Non-intrusive load disaggregation using graph signal processing," *IEEE Trans. Smart Grid*, vol. 9, no. 3, pp. 1739–1747, Aug. 2016.

[73] G.-Y. Lin, S.-C. Lee, Y.-J. Hsu, and W.-R. Jih, "Applying power meters for appliance recognition on the electric panel," in *Proc. 2010 IEEE Conf. Ind. Electron. Appl.*, Taichung, Taiwan, June 2010, pp. 2254–2259.

[74] J. Liao, G. Elafoudi, L. Stankovic, and V. Stankovic, "Non-intrusive appliance load monitoring using low-resolution smart meter data," in *Proc. IEEE SmartGridComm'14*, Venice, Italy, Jan. 2014, pp. 535–540.

[75] Y.-H. Lin and M.-S. Tsai, "Non-intrusive load monitoring by novel neuro-fuzzy classification considering uncertainties," *IEEE Trans. Smart Grid*, vol. 5, no. 5, pp. 2376–2384, Aug. 2014.

[76] M. B. Figueiredo, A. De Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (NILM) systems," in *Proc. 2011 Int. Conf. Adaptive Natural Comput. Algorithms*, Ljubljana, Slovenia, Apr. 2011, pp. 31–40.

[77] S. M. Tabatabaei, S. Dick, and W. Xu, "Toward non-intrusive load monitoring via multi-label classification," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 26–40, June 2016.

[78] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key technologies and open issues," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3072–3108, Fourth Quarter 2019.

[79] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, "Sequence-to-point learning with neural networks for non-intrusive load monitoring," in *Proc. AAAI 2018*, New Orleans, LA, Feb. 2018, pp. 1–8.

[80] C. Shin, S. Joo, J. Yim, H. Lee, T. Moon, and W. Rhee, "Subtask gated networks for non-intrusive load monitoring," in *Proc. AAAI 2019*, vol. 33, Honolulu, HI, Jan. 2019, pp. 1150–1157.

[81] K. Chen, Y. Zhang, Q. Wang, J. Hu, H. Fan, and J. He, "Scale-and context-aware convolutional non-intrusive load monitoring," *IEEE Trans. Power Syst.*, vol. 35, no. 3, pp. 2362–2373, Nov. 2019.

[82] M. Kaselimi, N. Doulamis, A. Voulodimos, E. Protopapadakis, and A. Doulamis, "Context aware energy disaggregation using adaptive bidirectional LSTM models," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3054–3067, July 2020.

[83] J. Kelly and W. Knottenbelt, "Neural NILM: Deep neural networks applied to energy disaggregation," in *Proc. ACM BuildSys'15*, Seoul, South Korea, Nov. 2015, pp. 55–64.

[84] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NIPS 2014*, Montreal, CA, Dec. 2014, pp. 3320–3328.

[85] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. ICML 2015*, vol. 37, Lille, UK, July 2015, pp. 97–105.

[86] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, Mar. 2018. [Online]. Available: https://arxiv.org/abs/1802.05365

[87] P. P. M. do Nascimento, "Applications of deep learning techniques on NILM," PhD Dissertation, Universidade Federal do Rio de Janeiro, 2016.

[88] A. M. Ahmed, Y. Zhang, and F. Eliassen, "Generative adversarial networks and transfer learning for non-intrusive load monitoring in smart grids," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Tempe, AZ, Nov. 2020, pp. 1–7.

[89] Y. Liu, L. Zhong, J. Qiu, J. Lu, and W. Wang, "Unsupervised domain adaptation for non-intrusive load monitoring via adversarial and joint adaptation network," *IEEE Transactions on Industrial Informatics*, Mar. 2021.

[90] Y. Liu, X. Wang, and W. You, "Non-intrusive load monitoring by voltage–current trajectory enabled transfer learning," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 5609–5619, Sept. 2019.

[91] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," *arXiv preprint arXiv:1703.03400*, July 2017. [Online]. Available: https://arxiv.org/pdf/1703.03400

[92] N. Batra, R. Kukunuri, A. Pandey, R. Malakar, R. Kumar, O. Krystalakos, M. Zhong, P. Meira, and O. Parson, "Towards reproducible state-of-the-art energy disaggregation," in *Proc. ACM BuildSys 2019*, New York City, NY, Nov. 2019, pp. 193–202.

[93] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, June 2014.

[94] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv preprint arXiv:2004.05439*, Apr. 2020. [Online]. Available: https://arxiv.org/abs/2004.05439

[95] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT 2010*, Sept. 2010, pp. 177–186.

[96] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, Feb. 2017. [Online]. Available: https://arxiv.org/abs/1707.03141

[97] W. Li, T. Logenthiran, V.-T. Phan, and W. L. Woo, "Implemented iot-based self-learning home management system (shms) for singapore," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2212–2219, June 2018.

[98] L. Wang, S. Mao, B. Wilamowski, and R. Nelms, "Pre-trained models for non-intrusive appliance load monitoring," *IEEE Transactions on Green Communications and Networking*, 2021, to appear.

[99] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[100] S. Duan, W. Yang, X. Wang, S. Mao, and Y. Zhang, "Temperature forecasting for stored grain: A deep spatio-temporal attention approach," *IEEE Internet of Things Journal*.

[101] Z. Yue, C. R. Witzig, D. Jorde, and H.-A. Jacobsen, "Bert4nilm: A bidirectional transformer model for non-intrusive load monitoring," in *Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring*, New York, NY, Nov. 2020, pp. 89–93.

[102] N. Lin, B. Zhou, G. Yang, and S. Ma, "Multi-head attention networks for nonintrusive load monitoring," in *2020 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, Macau, China, Aug. 2020, pp. 1–5.

[103] C. Wu, F. Wu, T. Qi, and Y. Huang, "Fastformer: Additive attention can be all you need," *arXiv preprint arXiv:2108.09084*, 2021.

[104] P. Huber, A. Calatroni, A. Rumsch, and A. Paice, "Review on deep neural networks applied to low-frequency nilm," *Energies*, vol. 14, no. 9, p. 2390, Apr. 2021.

[105] K. Chen, Q. Wang, Z. He, K. Chen, J. Hu, and J. He, "Convolutional sequence to sequence non-intrusive load monitoring," *The Journal of Engineering*, vol. 2018, no. 17, pp. 1860–1864, Nov. 2018.

[106] L. Mauch and B. Yang, "A new approach for supervised power disaggregation by using a deep recurrent LSTM network," in *Proc. IEEE GlobeSIP 2015*, Orlando, FL, Dec. 2015, pp. 63–67.

[107] J. Kim, T.-T.-H. Le, and H. Kim, "Nonintrusive load monitoring based on advanced deep learning and novel signature," *Computational Intelligence and Neuroscience*, vol. 2017, p. Article ID 4216281, Oct. 2017.

[108] R. Bonfigli, A. Felicetti, E. Principi, M. Fagiani, S. Squartini, and F. Piazza, "Denoising autoencoders for non-intrusive load monitoring: Improvements and comparative evaluation," *Elsevier Energy and Buildings*, vol. 158, no. 1, pp. 1461–1474, Jan. 2018.

[109] I. J. Goodfellow, "On distinguishability criteria for estimating generative models," *arXiv preprint arXiv:1412.6515*, 2014.

[110] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, Dec. 2016.

[111] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[112] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[113] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *arXiv preprint arXiv:2103.14030*, 2021.

[114] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[115] N. Tang, S. Mao, and R. M. Nelms, "Adversarial attacks to solar power forecast," in *Proc. IEEE GLOBECOM 2021*, Madrid, Spain, Dec. 2021.

[116] T. Zhang and S. Mao, "An introduction to the federated learning standard," *ACM GetMobile*, vol. 25, no. 3, pp. 18–, Sept. 2021.