UNIVERSITY OF CALIFORNIA
SANTA CRUZ

DEEP LEARNING BASED LOAD FORECASTING AND MONITORING FOR ELECTRIC POWER SYSTEMS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Jing Xiong

December 2023

The Dissertation of Jing Xiong
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Deep Learning Based Load Forecasting and Monitoring for Electric Power Systems

by

Jing Xiong

The hidden information and characteristics embedded in electrical load profiles are indispensable for the effective planning and operation of power grids. Load forecasting plays a vitally important role in many applications for the electric industry, e.g., energy generation and transactions, load shedding and restoration, as well as infrastructure expansions. Based on historical data, accurate load forecasts provide a good reference for the needed load demand, which can increase the efficiency and revenues of the electricity generation and distribution companies. In parallel with load forecasting, load monitoring can identify various types and statuses of loads by disaggregating the total power consumption into individual appliance levels. A scientific procedure of load monitoring facilitates the establishment of user-profiles, power usage habits, and peak load shifting. This is beneficial for both the end-users and utilities, improving the overall efficiency of the power network.

This research encapsulates three pivotal works focusing on enhancing the accuracy and efficiency of predicting and monitoring electrical loads. The first work introduces an attention-based neural load forecasting model that utilizes an encoder-decoder

RNN and BiLSTM for dynamic feature selection and similar temporal information adaptively, showing superior performance in short-term load forecasting tasks. The second piece builds upon this, offering a unifying framework that integrates time-varying feature weighting, hierarchical temporal attention, and feature-reinforced error correction. This framework has shown outstanding efficacy in electric load forecasting on public datasets. The third work pivots to Non-Intrusive Load Monitoring (NILM). It introduces MATNilm, a multi-appliance-task framework, which efficiently disaggregates total power usage with limited labeled data. It employs a training-efficient sample augmentation (SA) scheme and a shared-hierarchical-split structure with a two-dimensional attention mechanism, enhancing the recognition of spatio-temporal correlations among all appliances. Collectively, these contributions underscore the integration of advanced deep learning techniques to improve the efficiency, reliability, and accuracy of load forecasting and monitoring in power systems.

Extensive numerical simulations reveal that our proposed load forecasting framework surpasses several existing forecasting methods. We emphasized the critical roles of both the feature weighting mechanism and the error correction module in securing this superior performance. In relation to the NILM task, even with just a single day's training data and limited appliance operation profiles, our SA algorithm demonstrates test performance comparable to models trained with a comprehensive dataset. In tandem with our proposed model structure, the simulation results highlight that our approach offers a notably enhanced performance against numerous baseline models, reducing relative errors by over 50% on average.

Acknowledgments

Embarking on the Ph.D. journey has been an intricate tapestry of challenges, discoveries, and unyielding perseverance. As I stand at the culmination of this journey, I feel a profound sense of gratitude toward those who have stood by me, guiding, supporting, and inspiring me through every step of this academic odyssey.

I would like to express my sincere gratitude to my advisor, Professor Yu Zhang, for his invaluable guidance and unwavering support throughout the process of completing this dissertation. Under his mentorship, I had the opportunity to delve deep into the intersection of deep learning and load forecasting and monitoring of power systems, which formed the foundation of my research. Professor Zhang's expertise, insightful perspectives, and unwavering belief in my abilities have been truly transformative. His constant encouragement and constructive feedback have not only shaped my academic pursuits but have also contributed to my growth as a confident researcher. I am truly inspired by Professor Zhang's passion for the subject and his unwavering commitment to academic excellence.

I am grateful to my committee members, Prof. Leila Parsa, Prof. Keith Corzine, and Prof. Patrick Mantey, for their valuable feedback, thorough reviews, and constructive suggestions. Their expertise and dedication to higher learning have significantly contributed to refining my thesis, and I am indebted to them for leaving an indelible mark on my academic journey.

My time at Argonne National Lab was an enriching experience, and I am particularly thankful to Dongbo Zhao and Tianqi Hong. Developing the Python power system transient simulation platform with them provided me with practical insights that significantly contributed to my research. Furthermore, I extend my gratitude to my Machine Learning Intern experience at Soley Therapeutics, where I worked under the guidance of Wenpei Liu. Prototyping the AWS ETL microscopy image pipeline and implementing an outsourced data labeling pipeline were pivotal experiences that enhanced my practical understanding.

The Energy, Optimization & Data Analytics Lab (eODAL) at the University of California, Santa Cruz, has been my academic home, and I am indebted to my lab mates: Shourya Bose, Sifat-E-Tanzim Chowdhury, Gabriel Intriago, and Kejun Chen. The bond between them, their ability to have meaningful conversations, and their collaborative brainstorming sessions were extremely valuable. Every moment spent in eODAL was a great opportunity for collaborative learning.

I am also immensely grateful to the University of California, Santa Cruz, and the Electrical and Computer Engineering department for providing me with an environment that fostered learning, innovation, and growth.

To my parents, Xuehai Xiong and Jianni Kong, words fall short of expressing my gratitude. My family has been a constant source of support, encouraging me to pursue my dreams and teaching me the importance of hard work, perseverance, and humility. Their love and support have been the foundation of my strength, guiding me not only in my academic pursuits but in all aspects of my life.

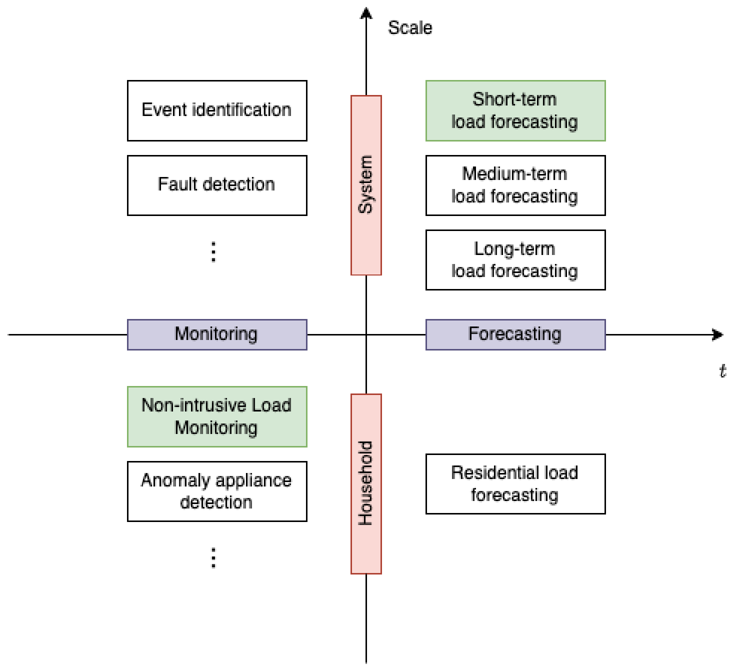
To my friends, Lastly, I would like to acknowledge the generous funding and support from various institutions and awards that made my Ph.D. research feasible. Their commitment to fostering research and innovation played a pivotal role in my academic journey.

Chapter 1

Introduction

The power system plays an indispensable role in society, serving as the foundation of a country's economy, security, and stability [1]. Consequently, enhancing the efficiency and robustness of the power grid remains a perennial goal for scholars. With the advent of the smart grid, the next-generation grid is expected to produce a massive amount of data each day. Data from the market, equipment, and system provides valuable information for the reliable and efficient planning and operation of power grids from SCADA and automatic meter readings. To extract meaningful insights from this vast data, data mining and analysis have emerged as primary research areas in the power and energy community.

Electrical load data represents one of the most prevalent measurements within power networks, providing crucial insights into energy generation and consumption across the entirety of the system. Figure 1.1 delineates the array of hidden information that can be gleaned from such load data, categorized by both temporal considerations



Scale
Event identification
Fault detection
⋮
System
Short-term load forecasting
Medium-term load forecasting
Long-term load forecasting
Monitoring
Forecasting
t
Non-intrusive Load Monitoring
Anomaly appliance detection
⋮
Household
Residential load forecasting

Figure 1.1: Examples of hidden information in load data in terms of time and scale level. The green box represents the tasks given by this research

and scale.

On the temporal front, load monitoring primarily centers around real-time or recent past data, aiming to capture the current state or immediate past behavior of the load. In contrast, load forecasting endeavors to anticipate future trends and patterns, helping operators and planners prepare for upcoming load scenarios.

When segmenting by scale, the load data can be broadly bifurcated into system-level and household-level data. The system-level pertains to broader infrastructure components like the transmission and distribution systems, encapsulating large-scale patterns and behaviors. The household-level, meanwhile, offers a more granular view, spotlighting individual residential consumption patterns and behaviors.

Load forecasting involves predicting future load behaviors based on historical load patterns and relevant features. At the household level, daily routines of residents dictate the load pattern, which often exhibits stochastic behavior and is challenging to predict [2]. As load data aggregates from a single household to the system level, the randomness diminishes, while the importance of precise forecasting amplifies. Considering that electrons travel at the speed of light and there is no practical solution yet for the mass storage of electricity, it is imperative to balance supply and demand at all times to ensure grid stability. System-level load forecasting offers advance notice of expected demand. Depending on the forecast horizon, load forecasting can be categorized into short (one hour to a week), medium (one week to a year), and long-term (one to twenty years) forecasting. Each caters to different applications and business needs. For instance, long-term forecasting is vital for power system planning, such as generation and transmission grid strategies. Medium-term forecasting aids in scheduling maintenance. In contrast, short-term load forecasting (STLF) is pivotal for day-ahead unit commitments, market clearances, reserve plans, energy bids, and economic load dispatches [3]. As the forecasting duration narrows, the demand for precise predictions intensifies. Although load forecasting has been a research focal point for years, recent incorporations of renewable energy source, energy storage systems (or facilities), and electric vehicle have reshaped user behaviors, posing considerable challenges to load demand prediction [4].

While forecasts provide anticipatory data, real-time load data unveils the actual state of a system. Load monitoring assists operators and consumers in comprehending the real-time status of systems or households, enabling their ability to respond to time-of-use (TOU) pricing, supply shortages, or outages.

There are two primary distinctions between load data at the system and household levels: the number of measurements and the granularity of data resolution. At the system level, there are multiple sources of measurements. The SCADA system provides a wide range of measurements, continuously monitoring the overall system status and samples every few seconds. Additionally, the PMUs can measure both the magnitude and phase angle of voltage and current, achieving up to 120 samples per second. Conversely, at the household level, the data is limited to aggregated power consumption. Smart meters are primarily designed to measure cumulative energy consumption and typically do not record the energy usage of individual appliances. Although these meters usually compute power at a 1Hz rate, utilities collect energy from the meter over 15-minute intervals, primarily for billing purposes. Devices such as Vue or Eagle [5] can extract data from the meter at a 1Hz frequency.

Under the smart grid paradigm, understanding customers' power consumption habits and offering guidance is invaluable. Customers can contribute to power generation through renewable sources and enhance grid efficiency by modifying their consumption habits, thereby leveling the grid's load curve and easing its load pressure. Although it's feasible to monitor household power consumption at frequent intervals, obtaining appliance-specific data is both costly and cumbersome. Non-intrusive load monitoring (NILM), or energy disaggregation, seeks to deduce individual appliance usage from a household's cumulative power signal without necessitating dedicated measurement devices for each appliance.

In recent years, deep neural networks (DNNs) have demonstrated their prowess in discerning intricate input-output relationships across various domains, including natural language processing [6] and computer vision [7]. These networks offer innovative solutions within the power system sector. This research endeavors to extract the latent information within load data via deep learning methodologies, aiming to refine the precision of load prediction and monitoring.

The rest of this thesis is structured as follows: Chapter 2 covers the literature review of short-term load forecasting and non-intrusive load monitoring. Following that, Chapter 3 introduces an attention-based neural load forecasting approach. In Chapter 4, a comprehensive deep learning framework for multi-horizon STLF is developed. Chapter 5 presents a solution for the NILM problem with limited training data. Lastly, Chapter 6 discusses future work.

Chapter 2

Literature Review

2.1 Short-term Load Forecasting

Load forecasting refers to the prediction of future load behavior derived from the historical load pattern and its relevant features. Based on different forecast horizons, load forecasting can be divided into three categories: short-term (one hour to a week), medium-term (one week to a year), and long-term (one to twenty years) forecasting. Each provides benefits for various applications and business needs. Long-term forecasting is mainly used in power system planning, such as generation and transmission expansion planning. Medium-term forecasting plays a crucial role in maintenance scheduling. Short-term load forecasting (STLF) is indispensable for day-ahead unit commitment, market clearing, spinning reserve plans, energy bidding, as well as economic load dispatch [3].

As the forecasting time span shrinks, the requirement for forecasting accuracy increases. This is due to the need for more precise predictions in a shorter time frame to effectively manage and distribute energy resources. In addition, wide applications of renewable energy generation, energy storage systems (or facilities), and electric vehicle in recent years have had a huge impact on users' load behavior. These pose significant challenges in forecasting load demand [4]. Various approaches have been proposed to improve the STLF accuracy. They can be roughly categorized into three classes: (i) time series analysis, (ii) classical machine learning algorithms, and (iii) deep learning models.

Time series analysis methods were widely studied in the late 20th century. Methods such as autoregressive (AR) [8], autoregressive moving average (ARMA) [9], and later on autoregressive integrated moving average (ARIMA) [10] were implemented for STLF. These methods are easy to implement and interpret; however, they require meticulous preprocessing to make a time series stationary [11]. Moreover, time series approaches are sensitive to irrelevant features and may fail to capture a long-term dependency.

After that, with the development of classical machine learning theory, researchers began to explore its application in STLF. Ceperic *et al.* proposed a support vector regression machines (SVR) approach for STLF, which minimizes the user interaction requirement by an adaptive model building strategy [12]. A random forest (RF) model was used for STLF, which could deal with nonstationarity, heteroscedasticity, trend, and multiple seasonal cycles load data [13]. In order to avoid information loss, Cheng *et al.* used different feature sets to construct an ensemble random forest-based model, which could also avoid the curse of dimensionality [14]. Taieb *et al.* implemented component-wise gradient boosting models (GBM) for each hour for multi-step short-term load forecasting [15]. These classic machine learning models can adapt to irrelevant features and are more robust in capturing the nonlinear behaviors of electricity load. However, most of these approaches use predetermined nonlinear models, which may prevent them from learning the true underlying nonlinearity effectively [16].

Over the past decade, deep neural networks (DNNs) have convincingly shown their capability to uncover the input-output relationships of different complex learning tasks in natural language processing [6] and computer vision [7]. Among different structures of DNNs, recurrent neural networks (RNNs) have been widely adopted and with proven efficacy in sequence-to-sequence learning tasks as well as in time-series forecasting tasks [17]. Despite the effectiveness of RNNs in time-series forecasting learning, a significant limitation of the plain RNNs is the well-known gradient vanishing problem [18]. Hochreiter and Schmidhuber proposed long short-term memory networks (LSTM) with a "three gates" technique to address this problem. The three gates: input gate, forget gate, and output gate, are designed to control information flows, and the relevant information is kept for long-term memory through these gates while the other information is forced to be ignored [19]. Later, in 2014, Cho *et al.* introduced gated recurrent units (GRUs), also a gating mechanism-based RNN. GRUs decrease the size of parameters by decreasing the number of gates and also reduce the training effort [20].

Sequence-to-sequence (seq2seq) learning [21] is one of the tasks that resolve the mapping between the sequential inputs and outputs of the task, which shares various similarities with time-series learning problems. The encoder-decoder structure usually serves as the backbone for most seq2seq models [22, 23]. Specifically, in time-series tasks, the encoder encodes the historical input features sequence into a single fixed-length vector, based on which the decoder yields the output. However, coping with a long input series is a challenge for the structure. To bypass this limitation, the attention mechanism was introduced to search for a set of positions in historical time steps where the most relevant information is concentrated [24]. For this new paradigm, a context vector is designed to bridge the gap between the encoder and the decoder, filtered for each output time step.

Although literature studies have shown that deep learning methods generally have better performance, single-level networks are challenging to deal with different data set conditions. In order to further improve the prediction accuracy, more modules can be added to the pipeline to combine the advantages of each module. Especially for STLF, feature engineering and error correction modules are necessary.

Rather than selecting a subset of features, feature weighting attempts to weight each feature based on their importance or relevance with the output [25]. Utilizing the feature weights given by the random forest, Xuan proposed a multi-model fusion based deep neural network to forecast the load demand [26]. Qin *et al.* proposed an input attention layer as feature weighting that can be trained simultaneously with the model [16]. However, their scheme is based only on the past information which cannot capture all the information of the entire input sequence. Moreover, the feature weighting part is embedded in the encoder, which makes it hard to be adopted for other basic structures;

e.g. the convolutional neural network (CNN).

The prediction error generally comes from two parts: the learning ability of the original model and the newly emerging unknown data. In order to further improve the prediction accuracy, the error correction module can learn useful hidden information from the error value to reduce the impact of the two parts mentioned above. Deng *et al.* proposed a hybrid model that includes a decomposition module, a forecasting module, and an error correction module for wind speed forecasting. Wind speed is first decomposed with empirical wavelet transform into several subseries. Then, each subseries is forecasted via an Elman neural network. Lastly, an ARIMA model is employed to forecast the error of each subseries and update the final output [27]. Inspired by the dynamic mode decomposition (DMD) method in fluid dynamics, Kong *et al.* used the DMD algorithm to capture the potential spatiotemporal dynamics of error series for STLF. This algorithm first constructs the error Hankel matrix and then does the pattern decomposition of the error. After that, an error series was reconstructed and finally forecasting the error [4]. The existing approach for an error correction system is usually to design a completely new model such as ARIMA [28] [27] or ELM [29] [30] to forecast the error. The new model will face the following shortcomings: 1) Increase the cost of learning. Model selection and hyperparameter tuning are inevitable, which will greatly increase design complexity and time usage. In addition, training a new model from scratch requires a large amount of data and is time-consuming, especially for deep-learning models. 2) Loss of the existing knowledge learned by the original model. This is even more critical. The original forecasting model learned some useful hidden patterns that could be useful for forecasting errors. The existing error correction system only deals with the error sequence and ignores the useful hidden knowledge learned by the predictive model.

In order to solve the above problems, the transfer learning approach came into being. Transfer learning was first discussed in the “Learning to Learn” workshop held by Neural Information Processing Systems (NIPS) in 1995. The motivation of transfer learning is to use previously acquired domain knowledge to solve new problems faster or with better solutions [31]. In recent years, transfer learning has been successfully used for classification, regression, and clustering problems. In load forecasting, researchers have also explored the use of transfer learning. In this context, they explore the way that the knowledge is transferred from one region/household to another region/household [32] [33] [34]. In this case, the source and target domains are the same, which are load and relative features, and the task is also the same, which is load forecasting. The key challenge for applying transfer learning in error correction is incorporating the error information into the target domain without changing the input dimension, while capitalizing on previously acquired feature knowledge.

2.2 Non-Intrusive Load Monitoring

Exploring users’ power usage patterns and providing energy saving guidance is important for smart power grids. Users can participate in power production activities through renewable energy generation and effectively improve power efficiency by adjusting power usage patterns to shift the peak load demand [35]. It is relatively easy to install a meter that monitors the household power consumption every few seconds, or at a higher sampling rate. However, recording the power usage of each appliance involves dedicated monitoring instrumentation for each appliance, which is impractical and expensive. In this context, non-intrusive load monitoring (NILM), also known as energy disaggregation, aims to leverage a household's aggregate power consumption signal to make inferences about individual appliance usage without using extra sensors. Based on the measurement sampling rate, NILM can be divided into two categories. High-frequency measurements are in the order of 10 kHz, including various features such as harmonics [36] and V-I trajectory [37, 38]. This type of data requires sophisticated and prohibitively expensive instrumentation hardware and is very costly. NILM via low-frequency data is a more challenging task. As the sampling rate drops below 1 Hz, distinct features used in the high-frequency data are no longer available. Therefore, extracting high-level hidden features is the key to achieving better performance.

NILM was first studied by Hart in 1992 [39]. Afterward, many classical models were developed to deal with the problem. [40] proposes the factorial hidden Markov model (FHMM). Based on that work, researchers design a conditional factorial hidden semi-Markov model that can incorporate extra features about the appliance status in [41]. More classical models are summarized in review papers [42–44].

Over the past few years, deep neural networks (DNNs) have demonstrated remarkable achievements in various fields, such as natural language processing and computer vision. Several studies in NILM have also delved into the utilization of DNNs, as highlighted in the work by Bousbiat et al. [45]. Classic DNN layers such as convolutional neural network (CNN) [46], recurrent neural network (RNN) [47] and temporal convolutional network (TCN) [48] have been investigated extensively. In [49], the authors propose a bidirectional long short-term memory (LSTM) model to capture the context information of the aggregated consumption. A Bayesian-optimized framework is utilized to select the best configuration of the proposed regression model. In [50], a parallel CNN and a bidirectional gated recurrent unit network are developed to extract spatial-temporal features whose weights are updated through an attention mechanism. In [51], the NILM task is modeled as a multiple-instance learning problem, and a convolutional recurrent neural network is trained with weak supervision.

An increasing number of works have investigated advanced DNNs for NILM. In [52], the authors develop a subtask gated network (SGN). The regression subnetwork outputs the electricity usage while the classification counterpart, which is used as a gating unit, specifies the on/off state for each given appliance. Based on this work, [53] enhances the model's effectiveness by incorporating an attention mechanism into the regression subnetwork. This integration intensifies the network's ability to identify and focus on the most pertinent parts of the input sequence, thereby amplifying its representational prowess. [54] develops an architecture using different receptive field size branches. Branch-wise gates connect corresponding branches in two subnetworks. To improve the performance, the authors employ a Wasserstein generative adversarial network (WGAN) with gradient penalty during the model's training. Other GAN structures are also explored. [55] proposes an EnerGAN++ structure for the NILM task. An autoencoder is implemented as the GAN generator, enabling a nonlinear signal source separation. The aggregate input concatenating with actual or estimated appliance consumption is the discriminator's input. In [56], the adversarial training process and the joint adaptation loss are introduced to leverage the hidden information from unlabeled data.

2.2.1 Single-appliance vis-à-vis Multi-appliance

There are different setups for NILM, which can be categorized based on the learning tasks. Typically, the regression task involves the inference of individual appliance's power usage. The classification task aims to determine the appliance's on/off status or different states related to power levels [57]. As summarized in the previous section, various machine learning approaches have been proposed for both tasks. In addition, tools such as pattern matching [58] and source separation [59] can also be found in the literature; see details in [60].

Given an aggregate power consumption signal, a learning model can be trained for each appliance separately, or it can be used to estimate the power signals for all appliances in consideration. We use the terms "single-appliance" and "multi-appliance" to refer to those two approaches, respectively. Most of the existing regression models adopt the single-appliance approach, which ignores the interactions among different appliances. For example, it is uncommon to operate a dishwasher and a microwave at the same time. The use of a dryer often comes after a washer. In contrast, such context-aware information can be implicitly incorporated by the multi-appliance approach into the modeling process, which can improve the disaggregation accuracy.

The multi-appliance approach in previous works focuses on the on/off status and state classification. For instance, a semi-supervised multi-label deep learning framework is proposed in [61] to monitor the on/off status of multiple appliances simultaneously. In [62], an HMM-based model is designed to infer appliance states over time by eliminating the possibility of unmodeled loads and outputting the used energy. A WaveNet, which is fed with active, reactive, and apparent power as inputs, is used to estimate appliance states and currents [57]. In addition, the total energy consumption of a set of appliances with similar operational patterns is estimated[63].

The direct regression task in multi-appliance setups is still an ongoing development, with research studies such as [52] and [64] highlighting its advantages over state classification. Direct regression allows for more accurate and detailed estimations for individual appliances. However, existing approaches have certain limitations, which include the vanilla neural network for multi-appliance (VMA) in [65], the U-Net architecture in [66], and the transfer learning techniques employed in [67]. These methods directly output power consumption for multiple appliances from the output layer without explicitly considering the inter-appliance relationships at different time steps. This limitation hinders their effectiveness in achieving more accurate estimations in real-world scenarios.

2.2.2 Data Augmentation

Data augmentation for improving NILM performance has been explored in the literature. [68] generates synthetic training data by randomly adding the appliance profile to the user's aggregate load. Besides the on-duration period of appliances, [69] also considers the off-duration when generating augmented synthetic data. However, it is hard to determine how much synthetic data is really needed. On the other hand, this type of data augmentation (DA) only incorporates existing appliance profiles, which are very limited compared with the potentially vast array of models of each appliance. In order to generate more appliance profiles, [54] proposes an on-state augmentation method by adding extra noise to the fridge power consumption signal and the corresponding aggregation input. Existing data augmentation algorithms are mainly designed for generating a synthetic dataset for each aggregate and appliance pair, which acts as a regularizer for each appliance separately, for increasing the generalization capability of the model [64]. Moreover, these methods are still trained by a full dataset, which may not fit the limited data scenario.

Some existing works also analyze the limited data scenario in the testing phase. In [70], the author compares the energy disaggregation performance with 100%, 50%, and 25% of each house data in the REDD dataset. In [54], the author shows the dishwasher and microwave performance of the proposed network with 20% of the training data and fridge with 5% and 10% of the training data. These works do not focus on data-limited scenarios, but validate the effects of data-limited scenarios on the proposed model. Therefore, there is a lack of sufficient mining of the conditions of limited data scenarios. In [71], a semi-supervised graph-based approach is proposed with a limited amount of ground-truth data and a large pool of unlabeled observations to classify the activation status of the appliances accurately. A novel LSTM combined with a probabilistic neural network (PNN) algorithm is proposed to classify the active appliance type in [72]. The authors leverage the transfer learning techniques to lower the requirements for training data further.

Chapter 3

Attention-based Neural Load Forecasting: A Dynamic Feature Selection Approach

Encoder-decoder-based recurrent neural networks (RNNs) have made significant progress in sequence-to-sequence learning tasks such as machine translation and conversational models. Recent works have shown the advantage of this type of network in dealing with various time series forecasting tasks. This chapter focuses on the problem of multi-horizon short-term load forecasting, which plays a key role in the power system's planning and operation. Leveraging the encoder-decoder RNN, an attention model is designed to select the relevant features and similar temporal information adaptively. First, input features are assigned with different weights by a feature selection attention layer, while the updated historical features are encoded by a bi-directional

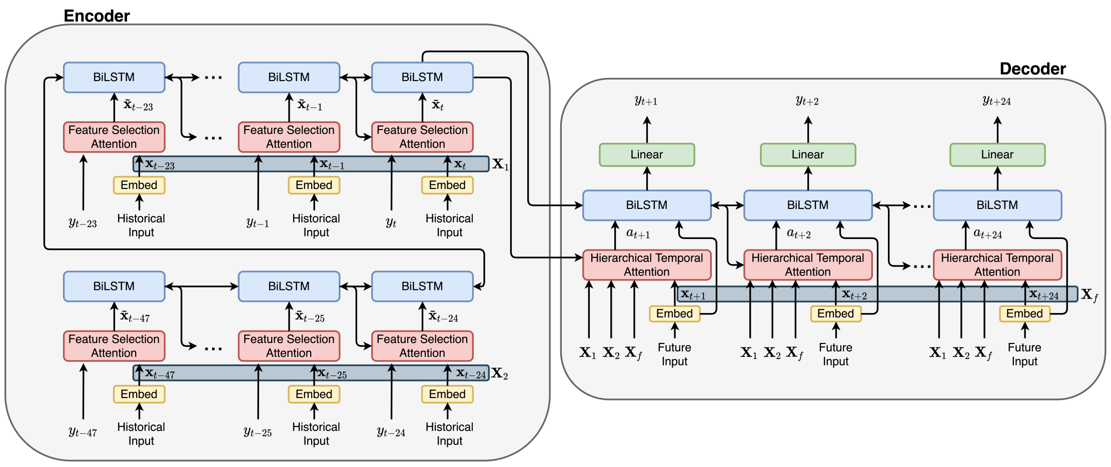


Figure 3.1: The network architecture of the proposed model. The feature selection attention in the encoder is designed to adaptively weigh the different input features. In contrast, the hierarchical temporal attention in the decoder focuses more on the temporal similarity to incorporate similar day information. To simplify the demonstration, the structure in the figure only reflects the scenario of forecasting the next day based on the previous two days. This can be easily extended to a general case.

long short-term memory (BiLSTM) layer. Then, a decoder with hierarchical temporal attention enables a similar day selection, which re-evaluates the importance of historical information at each time step.

3.1 Proposed Approach

The novel architecture of our proposed attention-based DNN model, whose backbone is a competitive encoder-decoder structure, is illustrated in Fig. 3.1. In this section, we will elaborate on the details of each module by highlighting the new design of dynamic feature selection and similar day selection for multi-horizon STLF.

3.1.1 Input Feature Embedding and Problem Statement

By transforming categorical data into a numeric vector, embedding is widely used in machine learning when inputs contain one or more discrete values or items from a finite set of choices. For STLF, the inputs can contain meteorological conditions (e.g., temperature, humidity, wind speed and direction, etc), indicator of holiday, and utility discount programs, e.g., TOU pricing. In this work, we simply use the one-hot encoding for categorical features. After embedding, the inputs of the encoder are the concatenation of embedded categorical features, continuous numeric features, and historical target values (active power demand) at each time step $t \in [t - T_h + 1, t]$, while inputs of the decoder are the concatenation of embedded categorical features, continuous numeric features at each time step $t \in [t + 1, t + T_f]$, and historical feature matrix $\mathbf{X}_i$ and future feature matrix $\mathbf{X}_f$, where T_h and T_f are the window size of historical and future data, respectively.

Let $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n) \in \mathbb{R}^{(T_h+T_f)\times n}$ collect all embedded input features across the entire horizon of interest. Thus, each row vector of $\mathbf{X}$ denoted as $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$ represents all n input features at time t. For the multi-horizon STLF task, given the historical feature inputs $\mathcal{X}_h := \{\mathbf{x}_h\}_{h=t}^{t-T_h+1}$ and the corresponding target values $\mathbf{y}_h = (y_t, y_{t-1}, \ldots, y_{t-T_h+1})^\top$, we aim to learn a function $f(\cdot)$ that can predict future target values $\mathbf{y}_f = (y_{t+1}, y_{t+2}, \ldots, y_{t+T_f})^\top$ for the given future features $\mathcal{X}_f := \{\mathbf{x}_f\}_{f=t+1}^{t+T_f}$; i.e., $\mathbf{y}_f = f(\mathcal{X}_f \mid \mathbf{y}_h, \mathcal{X}_h)$.

3.1.2 Encoder and Decoder

The encoder-decoder structure is a workhorse in state-of-the-art deep neural networks. For time series forecasting, the encoder maps historical input features $\mathbf{x}_t$ and output $y_{h,t}$ at each time step t to a hidden vector $\mathbf{h}_t$ that is passed to the decoder. Then, the decoder uses the last step hidden state of the encoder as its initial hidden state and outputs future target values based on future feature inputs.

3.1.2.1 Encoder with feature selection attention

As shown in Fig. 3.1, the encoder consists of two sublayers: a feature selection attention layer and a BiLSTM layer. The time range for the encoder is from time $t - T_h + 1$ to t.

Feature selection is key for almost all machine learning tasks. Irrelevant features can significantly affect the model's performance. By using feature selection attention, the proposed model is able to adaptively weigh different features and give more attention to features that contribute more to the target values. The weight α_t^k of each feature k at time t is calculated via the softmax operator

$$\alpha_t^k = \frac{\exp\left(e_t^k\right)}{\sum_{i=1}^n \exp\left(e_t^i\right)}, \; k = 1, 2, \ldots, n \tag{3.1}$$

where e_t^k is the k-th element of the vector $\mathbf{e}_t = (e_t^1, e_t^2, \ldots, e_t^n)^\top \in \mathbb{R}^n$ that is given by

$$\mathbf{e}_t = \mathbf{V}_e \tanh\left(\mathbf{W}_e \left[{\mathbf{h}_{t-1}^e}^\top; \mathbf{x}_t; y_t\right]^\top + \mathbf{b}_w\right) + \mathbf{b}_v. \tag{3.2}$$

The weight matrices $\mathbf{V}_e \in \mathbb{R}^{n\times p}$, $\mathbf{W}_e \in \mathbb{R}^{p\times(2hs+n+1)}$ and bias term $\mathbf{b}_w \in \mathbb{R}^{p\times 1}$, $\mathbf{b}_v \in \mathbb{R}^{n\times 1}$ are trained jointly with the proposed model. p is a hyper-parameter while hs is the hidden size of the BiLSTM. $[\mathbf{a};\mathbf{b}]$ denotes the concatenation of two row vectors $\mathbf{a}$ and $\mathbf{b}$. $\mathbf{h}^e_{t-1}$ is the hidden vector of the encoder BiLSTM at time $t-1$. Then, the weighted feature input is given by $\tilde{\mathbf{x}}_t = (\alpha^1_t x^1_t, \alpha^2_t x^2_t, \dots, \alpha^n_t x^n_t)$.

The BiLSTM consists of two LSTM layers in opposite directions which can capture the complete information of the entire input sequence at each time step. Let $\mathbf{h}^f_t$, $\mathbf{h}^b_t \in \mathbb{R}^{hs}$ denote the hidden state of forward and backward LSTM at time t, respectively. Given the weighted input $\tilde{\mathbf{x}}_t$, the hidden states of the encoder are updated iteratively from time $t - T_h + 1$ to t as follows [73]

$$\mathbf{h}^f_t = \mathrm{LSTM}^f(\mathbf{h}^f_{t-1}, \tilde{\mathbf{x}}_t) \tag{3.3a}$$

$$\mathbf{h}^b_t = \mathrm{LSTM}^b(\mathbf{h}^b_{t+1}, \tilde{\mathbf{x}}_t) \tag{3.3b}$$

$$\mathbf{h}^e_t = \left[{\mathbf{h}^f_t}^\top ; {\mathbf{h}^b_t}^\top\right]^\top . \tag{3.3c}$$

The LSTM$(\cdot)$ involves the following detailed updates [19]:

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{ix}\tilde{\mathbf{x}}_t^\top + \mathbf{b}_{ix} + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{b}_{ih}\right) \tag{3.4a}$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{fx}\tilde{\mathbf{x}}_t^\top + \mathbf{b}_{fx} + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{b}_{fh}\right) \tag{3.4b}$$

$$\mathbf{g}_t = \tanh\left(\mathbf{W}_{gx}\tilde{\mathbf{x}}_t^\top + \mathbf{b}_{gx} + \mathbf{W}_{gh}\mathbf{h}_{t-1} + \mathbf{b}_{gh}\right) \tag{3.4c}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{ox}\tilde{\mathbf{x}}_t^\top + \mathbf{b}_{ox} + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{b}_{oh}\right) \tag{3.4d}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{3.4e}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right), \tag{3.4f}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{g}_t$, $\mathbf{o}_t$ are the input, forget, cell, and output gates, respectively. $\mathbf{W}_{\cdot,x} \in \mathbb{R}^{hs\times n}$, $\mathbf{W}_{\cdot,h} \in \mathbb{R}^{hs\times hs}$ are the weight matrices; $\mathbf{b}_{\cdot,\cdot} \in \mathbb{R}^{hs}$ the bias vectors; $\mathbf{c}_t \in \mathbb{R}^{hs}$the cell state; $\sigma(\cdot)$ the sigmoid function; and $\odot$ the Hadamard product (element-wise multiplication).

3.1.2.2 Decoder with hierarchical temporal attention

The decoder leverages the encoder's last-step hidden state and cell state as its initial state. Using the last forecast value as the input for the next time step may introduce error propagation. Instead, we design a BiLSTM along with a feed-forward layer to predict the final target sequence once the BiLSTM learns all input information. The time range for the decoder is from time $t+1$ to $t+T_f$.

Incorporating the information of similar days and hours has been considered in the literature for load forecasting; see e.g., [74, 75]. However, such information is often treated as additional input features or used to generate separate models. In this section, we develop a novel hierarchical temporal attention layer, which incorporates a similar day soft selection to re-evaluate the importance of historical information at each time step t.

Consider using previous M days of historical data to forecast the hourly loads for the next day, where each day includes $t_d = 24$ data points. Thus, we have $T_h = M \times t_d$ and $T_f = t_d$. Let $\mathbf{X}_i = (\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^n) \in \mathbb{R}^{t_d \times n}$ and $\mathbf{X}_f = (\mathbf{x}_f^1, \mathbf{x}_f^2, \dots, \mathbf{x}_f^n) \in \mathbb{R}^{t_d \times n}$ collect the historical features for day i and the future features for the next day, respectively. We use the sum of feature-by-feature dissimilarities $D(\mathbf{X}_i, \mathbf{X}_f) = \sum_{k=1}^{n} \|\mathbf{x}_i^k - \mathbf{x}_f^k\|_2$ to quantify the distance between all features of those two days. Then, the similar day weight γ_i is calculated as the softmax of the reciprocal of the distance:

$$\gamma_i = \frac{\exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)}{\sum_{i=1}^{M} \exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)}, \; i = 1, 2, \dots, M. \tag{3.5}$$

When forecasting load at time t, not all historical data contribute equally to the model's output. Hence, the attention mechanism facilitates the extraction of historical information that is more important to the current forecast value. Let subscript i denote the i-th day and j for j-th hour. Then, the attention weight $\beta_{i,j,t}$ is given by

$$\beta_{i,j,t} = \frac{\exp(d_{i,j,t})}{\sum_{i=1}^{M} \sum_{j=1}^{t_d} \exp(d_{i,j,t})}, \tag{3.6}$$

where $d_{i,j,t}$ is the $(i \times t_d + j)$-th element of vector $\mathbf{d}_t = (d_t^1, d_t^2, \dots, d_t^{T_h})^\top \in \mathbb{R}^{T_h}$, which is given as

$$\mathbf{d}_t = \mathbf{V}_d \tanh\left(\mathbf{W}_d \left[{\mathbf{h}^d_{t-1}}^\top; \mathbf{x}_t\right]^\top + \mathbf{b}_{wd}\right) + \mathbf{b}_{vd}. \tag{3.7}$$

The parameters $\mathbf{V}_d \in \mathbb{R}^{T_h \times p'}$, $\mathbf{W}_d \in \mathbb{R}^{p' \times (2hs+n)}$, $\mathbf{b}_{vd} \in \mathbb{R}^{T_h \times 1}$ and $\mathbf{b}_{wd} \in \mathbb{R}^{p' \times 1}$ are trained jointly with the proposed model. $\mathbf{h}^d_{t-1}$ is the hidden vector of the decoder BiLSTM at time $t-1$.

To this end, let $\mathbf{h}_{i,j}$ denote the historical hidden state for the j-th hour in the i-th day from the encoder. The context vector of hierarchical temporal attention is calculated as $\mathbf{a}_t = \sum_{i=1}^{M}\sum_{j=1}^{t_d} \gamma_i \beta_{i,j,t} \mathbf{h}_{i,j}$. Given the future feature input $\mathbf{x}_t$ and context vector of hierarchical temporal attention $\mathbf{a}_t$, the hidden state of the decoder is updated iteratively from time $t+1$ to $t+T_f$ as $\mathbf{h}^d_t = \text{BiLSTM}(\mathbf{h}^d_{t-1}, \mathbf{h}^d_{t+1}, [\mathbf{x}_t; \mathbf{a}_t^\top])$.

Finally, a fully connected layer with the rectified linear unit (ReLU) activation function is used to transform the hidden information to the forecast output

$$\mathbf{y}_f = \mathbf{V}_y \text{ReLU}(\mathbf{W}_y \mathbf{h}^d_{t+1:t+T_f} + \mathbf{b}_{wy}) + \mathbf{b}_{vy}, \tag{3.8}$$

where $\mathbf{V}_y \in \mathbb{R}^{T_f \times p''}$ and $\mathbf{W}_y \in \mathbb{R}^{p'' \times (T_f \times 2hs)}$ are weight matrices; $\mathbf{b}_{vy} \in \mathbb{R}^{T_f \times 1}$ and $\mathbf{b}_{wy} \in \mathbb{R}^{p'' \times 1}$ are bias term; and vector $\mathbf{h}^d_{t+1:t+T_f} = \left[{\mathbf{h}^d_{t+1}}^\top; {\mathbf{h}^d_{t+2}}^\top; \ldots; {\mathbf{h}^d_{t+T_f}}^\top\right]^\top$. In the decoder module, both p' and p'' are hyper-parameters to be tuned.

Table 3.1: Inputs of the proposed model.

Input	Size	Description
$\mathbf{y}_h$	168×1	Historical target values
Temperature	192×1	Historical and future temperature
Holiday	192×1	Holiday and non-holiday indicators
Hour of Day	192×24	One-hot encoding
Day of Week	192×7	One-hot encoding
Month of Year	192×12	One-hot encoding

3.2 Numerical Results

3.2.1 Data Description

We evaluated the proposed model on an hourly load data introduced by the Global Energy Forecasting Competition 2014 extended (GEFCom2014-E) dataset [76]. The dataset includes load demand, temperature, and timestamps across a total of 9 years between January 1, 2006 to December 31, 2014. We use the data from the year 2010 to 2012 for training, the year 2013 for validation, and the year 2014 for testing. The data of the past seven days are used to forecast the next day. Specifically, the inputs of the model are listed in Table 3.1. The dataset is first standardized (subtract the mean and divide by the standard deviation) to mitigate the magnitude impact of different input features.

3.2.2 Baseline and Model Setup

To show the effectiveness of our proposed model, we compared two different types of models: classical machine learning models and LSTM based models. We implemented the Support Vector Regression (SVR), Random Forest (RF), and Gradient Boosting Machine(GBM) by using Scikit-Learn 0.23.2. For LSTM based methods, we considered the basic Encoder-Decoder based LSTM (EDLSTM), Encoder-Decoder based BiLSTM (EDBiLSTM) using bidirectional LSTM in the Decoder, proposed method with attention only applied in the encoder (eAttention), proposed method with attention only applied in the decoder (dAttention) and proposed method with feature weight calculated by Random Forest [77] (RFAttention). All the LSTM based methods are trained by the Adam optimizer with the mean squared error (MSE) loss function, which are implemented via PyTorch 1.6.0. To ensure fair comparisons, all models share the same training, validation, and testing datasets as well as input features. The grid search is used to tune the hyper-parameters. The final setting are listed as follows:

- SVR: linear kernel with $C = 0.1$.
- RF: number of estimators $= 1,000$, max-depth $= 20$, min-samples-split $= 2$, min-samples-leaf $= 1$.
- GBM: loss = "ls", learning rate $= 0.01$, number of estimators $= 1,000$, max-depth $= 5$, min-samples-split $= 2$, min-samples-leaf $= 15$.
- EDLSTM: batch $= 64$, hidden size $= 1,024$, epochs $= 5$.
- EDBiLSTM: batch $= 64$, hidden size $= 1,024$, epochs $= 5$.
- RFAttention: batch $= 128$, hidden size $= 256$, epochs $= 15$.

- dAttention: batch = 256, hidden size = 128, epochs = 5.
- eAttention: batch = 256, hidden size = 128, epochs = 20.
- ANLF: batch = 128, hidden size = 256, epochs = 5.

3.2.3 Numerical Results

Table 3.2 presents a comprehensive assessment of the performance of nine competing models over the entire testing period. Using four error metrics, namely mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE), and normalized root mean square error (nRMSE) [78], the evaluation offers a detailed insight into each model's forecasting capabilities. GBM is considered to be one of the best performers among classical machine learning models, with a performance that is comparable to dAttention and EDBiLSTM. A deeper dive into LSTM-based models reveals that incorporating attention mechanisms into both the encoder and decoder results in a marked improvement in accuracy. This is evident when comparing the likes of EDLSTM to its attention-enhanced counterparts. The results associated with RFAttention and ANLF underscore the efficacy of the proposed feature selection layer. Distinguished from its peers, the ANLF model, with its encoder-decoder RNN architecture, sophisticated attention mechanisms, and the capability to assign differential weights to input features coupled with bi-directional LSTM, excels in capturing intricate patterns, leading to unparalleled forecasting precision. Undoubtedly, the ANLF approach stands out, substantially surpassing all other baseline models in performance.

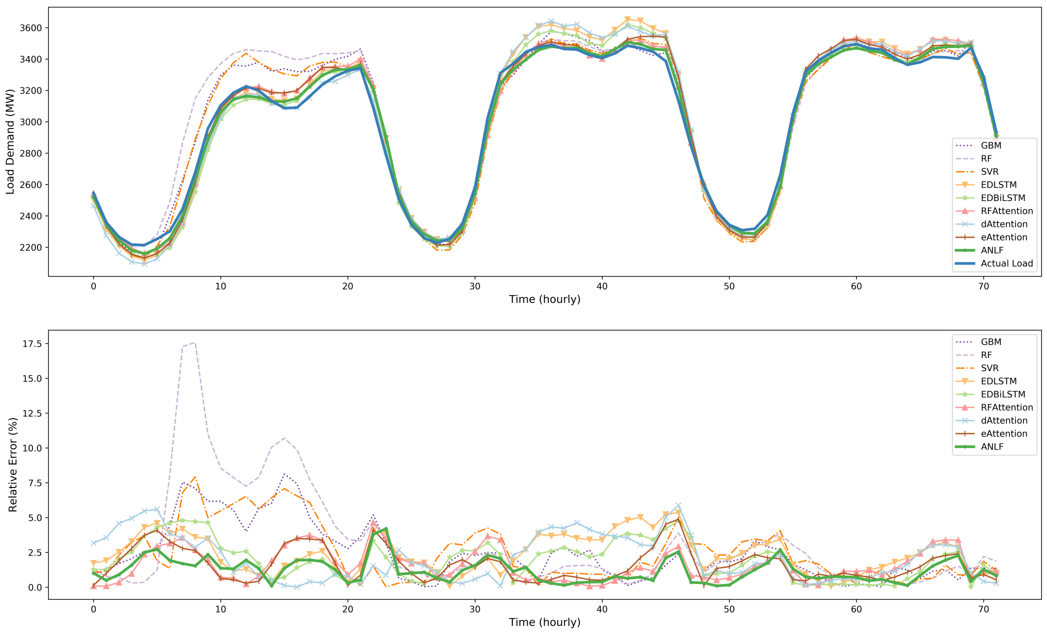


Figure 3.2: The detailed forecasting performance of nine different models over three days.

The effectiveness of this approach is highlighted by the simulation results, which demonstrate the significance of the attention mechanism in identifying relevant features and corresponding temporal data for the STLF task. A granular 3-day forecasting performance is delineated in Fig. 3.2. Here, the relative error between the forecasted value $\hat{y}$ and the actual value y is articulated as RE $= \frac{|y-\hat{y}|}{y} \times 100\%$. Furthermore, Table 3.3 provides a comprehensive month-wise performance comparison of six prominent models. This comparison serves to highlight the consistent superiority of ANLF across different temporal scopes.

In the box plot (Figure 3.3), we compared the relative errors of each model for the test data in the year 2014, illustrating the distribution of relative errors. The blue box represents the interquartile range, indicating values between the 25th and 75th

Table 3.2: Forecasting Errors over the test year 2014.

Model	MAE	RMSE	MAPE (%)	nRMSE (%)
GBM	71.48	94.61	2.15	2.84
RF	85.36	113.67	2.55	3.42
SVR	79.23	108.27	2.33	3.26
EDLSTM	74.73	97.95	2.26	2.94
EDBiLSTM	71.48	92.94	2.17	2.79
RFAttention	73.72	97.83	2.23	2.95
dAttention	70.91	91.42	2.15	2.75
eAttention	65.33	87.85	1.97	2.65
ANLF	**64.80**	**86.74**	**1.93**	**2.61**

Table 3.3: Performance comparison of six models with respect to each month.

Month	ANLF	EDBiLSTM	EDLSTM	GBM	RF	SVR
1	2.15	2.72	2.17	2.62	3.36	**2.12**
2	**1.79**	2.39	2.34	2.08	2.56	2.42
3	**2.36**	3.84	3.93	2.81	3.50	2.57
4	**1.82**	2.24	2.77	2.36	2.89	2.24
5	1.83	1.88	2.00	**1.75**	1.99	1.89
6	**1.66**	2.01	1.79	1.79	2.05	2.22
7	1.96	**1.79**	2.00	1.88	2.32	2.77
8	**1.45**	1.50	1.46	1.76	2.08	2.32
9	**1.67**	2.00	2.05	1.95	2.25	2.49
10	**1.52**	1.74	1.70	1.79	1.78	1.71
11	2.55	**1.94**	2.57	2.87	3.39	2.64
12	2.44	**2.08**	2.37	2.21	2.66	2.51
Overall	**1.93**	2.17	2.26	2.15	2.55	2.33

percentiles, while the orange line indicates the median of the relative errors. The blue numbers denote the count of forecasting points with errors exceeding 10%. Notably, our proposed model only has 16 such points, significantly fewer than other models. This improvement can be attributed to the hierarchical temporal attention mechanism, as

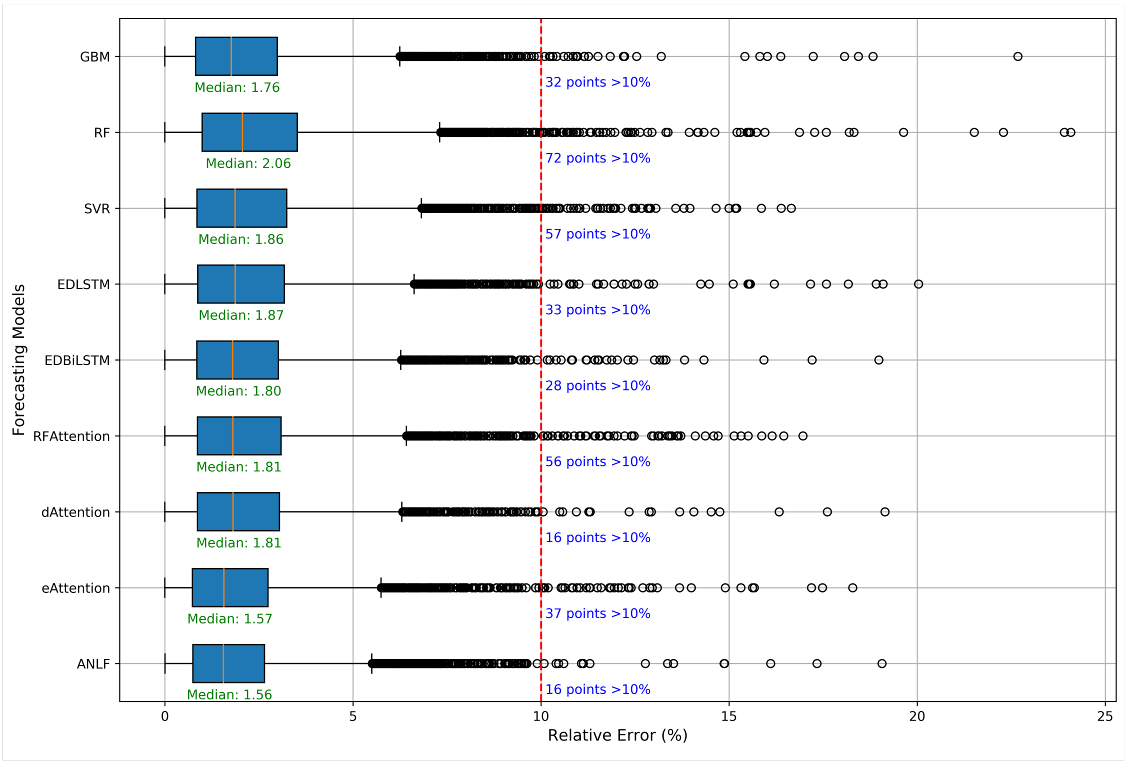


Figure 3.3: Box plot comparing the relative errors of each model

evidenced by our ablation study. Such performance further underscores the proposed model's capability to achieve a high level of accuracy and reliability.

3.3 Summary

In this chapter, we developed an end-to-end attention-based neural load forecasting framework for multi-horizon STLF. For the proposed approach, the dynamic feature selection layer combined with a BiLSTM encoder is designed to extract more relevant features. Then, a BiLSTM decoder with a hierarchical temporal attention layer decodes the next day load based on the future input features. The temporal attention layer provides a systemic way to incorporate similar day information. The extensive simulation results show the effectiveness of our proposed approach that has an edge over the state-of-the-art benchmarks.

Chapter 4

A Unifying Framework of Attention-based Neural Load Forecasting

Accurate load forecasting is critical for reliable and efficient planning and operation of electric power grids. In this chapter, a unifying deep learning framework is proposed for load forecasting, which includes time-varying feature weighting, hierarchical temporal attention, and feature-reinforced error correction. The framework adopts a modular design with good generalization capability. First, the feature-weighting mechanism assigns input features with temporal weights. Second, a recurrent encoder-decoder structure with hierarchical attention is developed as a load predictor. The hierarchical attention enables a similar day selection, which re-evaluates the importance of historical information at each time step. Third, an error correction module is developed that ex-

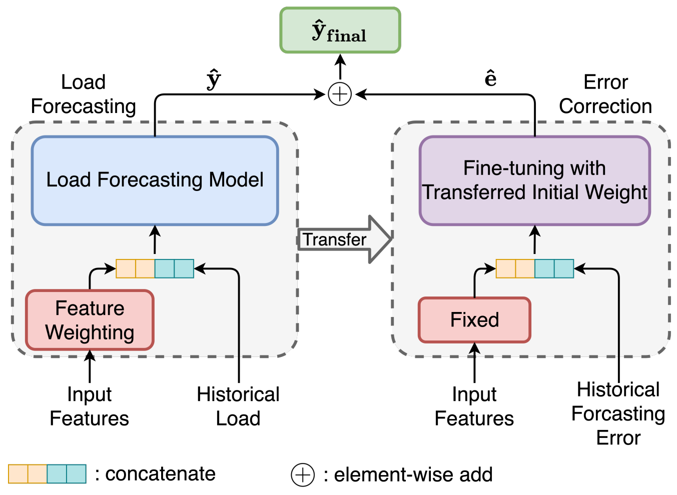


Figure 4.1: The architecture of the proposed framework. The left side is the load forecasting module with an input feature-weighting mechanism designed to weigh the different input features. The right side is the error correction module transferred from the left side model to further enhance forecasting ability. The detailed structures of feature-weighting mechanism and load forecasting module are shown in Fig. 4.2 and Fig. 4.3.

plores the errors and learned feature hidden information to further improve the model's forecasting performance. The framework provides an effective solution to the electric load forecasting problem, which can be further adapted to many other forecasting tasks.

4.1 The Proposed Load Forecasting Framework

In the following section, we introduce the overall load forecasting framework structure as shown in Fig. 4.1. The framework mainly consists of three modules: (i) the feature-weighting mechanism, (ii) the short-term load forecasting module, and (iii) the error correction module.

4.1.1 Feature Embedding and Feature-weighting Mechanism

Input features can generally be divided into two categories: numeric features and categorical features. As the model requires numeric input, a categorical feature would be transformed into a numeric vector. For STLF, the inputs can contain meteorological conditions (e.g., temperature, humidity, wind speed and direction, etc), time related features (e.g., indicator of holidays, season, etc), and utility discount programs e.g. TOU pricing. For those categorical features, we use the one-hot encoding in this work. After the embedding, all input features $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n) \in \mathbb{R}^{(T_h+T_f)\times n}$ are the concatenation of encoded categorical features and continuous numeric features. Each row of $\mathbf{X}$, denoted as $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)$, represents all features at time t.

Feature selection plays a crucial role in machine learning methods [79]. Irrelevant features can significantly affect the model performance. Instead of making a hard feature selection which is a special case of feature weighting mechanism, the proposed model is able to adaptively weigh different features and give more attention to features that contribute more to the target values. In [80], the feature selection layer is entangled in the encoder. Therefore, it is hard to transfer to other load forecasting modules. In order to modularize the proposed framework, we separate the feature selection layer from the encoder, and the weight α_t^k of each feature k at time t is calculated via the softmax operator:

$$\alpha_t^k = \frac{\exp\left(h_t^k\right)}{\sum_{i=1}^{n} \exp\left(h_t^i\right)},\ k = 1, 2, \ldots, n, \tag{4.1}$$

where h_t^k is the k-th entry of the vector $\mathbf{h}_t = \mathbf{V}_\alpha \tanh\left(\mathbf{W}_\alpha \mathbf{x}_t^\top + \mathbf{b}_{w\alpha}\right) + \mathbf{b}_{v\alpha} \in \mathbb{R}^n$.

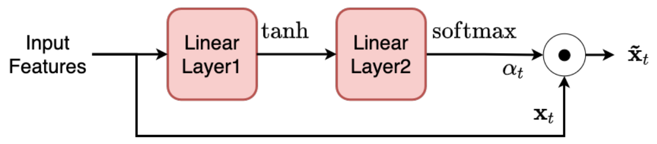


Figure 4.2: The feature-weighting mechanism structure with two linear layers.

The parameters $\mathbf{V}_\alpha \in \mathbb{R}^{n \times d_h^{fw}}$, $\mathbf{W}_\alpha \in \mathbb{R}^{d_h^{fw} \times n}$, $\mathbf{b}_{v\alpha} \in \mathbb{R}^{n \times 1}$, $\mathbf{b}_{w\alpha} \in \mathbb{R}^{d_h^{fw} \times 1}$ are trained jointly with the proposed model. d_h^{fw} is the number of neurons in the hidden layer and a hyper-parameter to tune. Then, the weighted feature input is given by $\tilde{\mathbf{x}}_t = \alpha_t \odot \mathbf{x}_t$, where $\odot$ denotes the element-wise multiplication. The detailed structure for the feature weighting mechanism is shown in Fig. 4.2.

The encoder's inputs are the concatenation of embedded categorical features, continuous numeric features, and historical target values (e.g. active power demand) at each time step $t \in [t - T_h + 1, t]$. The decoder's inputs are the concatenation of embedded categorical features and continuous numeric features at each time step $t \in [t+1, t+T_f]$. T_h and T_f are the window size of historical and future data, respectively.

4.1.2 Short-term Load Forecasting Model

4.1.2.1 Encoder-decoder structure

The encoder-decoder structure is a workhorse in state-of-the-art deep neural networks. For time series forecasting, the encoder maps historical input features $\mathbf{x}_t$ and output $y_{h,t}$ at each time step to a hidden vector $\mathbf{h}_t$ that is passed to the decoder. Then, the decoder uses the last step hidden state of the encoder as its initial hidden state, and outputs future target values based on future feature inputs. In this work, a

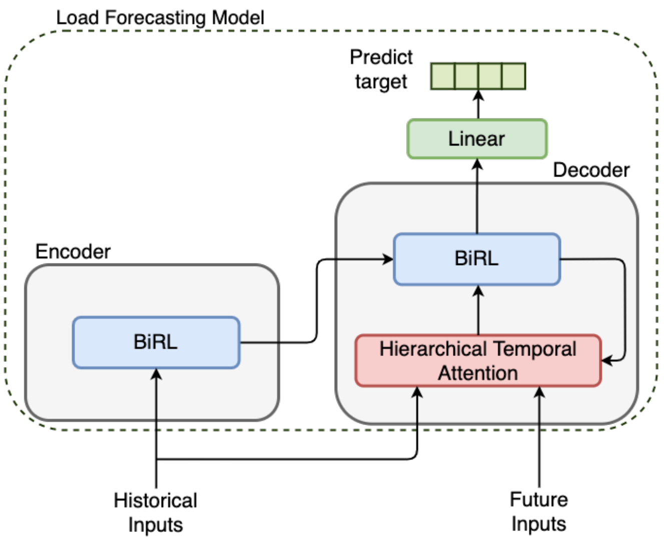


Figure 4.3: The network architecture of the load forecasting model. The hierarchical temporal attention in the decoder focuses on the temporal similarity to incorporate similar day information.

bi-directional recurrent layer (RL) is used for both encoder and decoder. We abbreviate the formulation for bi-directional RL as $\mathbf{h}_t = \text{BiRL}(\cdot)$, where RL can be chosen as recurrent neural network (RNN), LSTM or gated recurrent unit (GRU).

The BiRL consists of two sub-layers in opposite directions which can capture the complete information of the entire input sequence at each time step. Let $\mathbf{h}_t^f$, $\mathbf{h}_t^b \in \mathbb{R}^{hs}$ denote the hidden state of forward and backward recurrent layer at time t, respectively. Given a sequence of historical weighted feature and target value pairs

$(\tilde{\mathbf{x}}_t, y_t)$, the encoder's hidden states are updated from time $t - T_h + 1$ to t as

$$\mathbf{h}_t^f = \mathrm{RL}^f(\mathbf{h}_{t-1}^f, [\tilde{\mathbf{x}}_t; y_t]^\top), \tag{4.2a}$$

$$\mathbf{h}_t^b = \mathrm{RL}^b(\mathbf{h}_{t+1}^b, [\tilde{\mathbf{x}}_t; y_t]^\top), \tag{4.2b}$$

$$\mathbf{h}_t^e = \left[{\mathbf{h}_t^f}^\top; {\mathbf{h}_t^b}^\top\right]^\top, \tag{4.2c}$$

where $[\mathbf{a}; \mathbf{b}]$ denotes the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$.

Using future weighted feature $\tilde{\mathbf{x}}_t$ and context vector of hierarchical temporal attention $\mathbf{a}_t$ (see next subsection) as inputs, the decoder updates the hidden state iteratively from time $t+1$ to $t+T_f$ with initial state $\mathbf{h}_t^e$. Hence, we have $\mathbf{h}_t^d = \mathrm{BiRL}(\mathbf{h}_t^e, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top])$ with the detailed steps as

$$\mathbf{h}_t^f = \mathrm{RL}^f(\mathbf{h}_{t-1}^f, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top]), \tag{4.3a}$$

$$\mathbf{h}_t^b = \mathrm{RL}^b(\mathbf{h}_{t+1}^b, [\tilde{\mathbf{x}}_t; \mathbf{a}_t^\top]), \tag{4.3b}$$

$$\mathbf{h}_t^d = \left[{\mathbf{h}_t^f}^\top; {\mathbf{h}_t^b}^\top\right]^\top. \tag{4.3c}$$

Finally, a fully connected layer with the rectified linear unit (ReLU) activation function is used to transform the hidden information to the forecast output:

$$\mathbf{y}_f = \mathbf{V}_y \mathrm{ReLU}(\mathbf{W}_y \mathbf{h}_{t+1:t+T_f}^d + \mathbf{b}_{wy}) + \mathbf{b}_{vy}, \tag{4.4}$$

where $\mathbf{V}_y \in \mathbb{R}^{T_f \times d_h^o}$ and $\mathbf{W}_y \in \mathbb{R}^{d_h^o \times (T_f \times 2hs)}$ are weight matrices, $\mathbf{b}_{vy} \in \mathbb{R}^{T_f \times 1}$ and

$\mathbf{b}_{wy} \in \mathbb{R}^{d_h^o \times 1}$ are bias term, and $\mathbf{h}^d_{t+1:t+T_f} = \left[\mathbf{h}^{d\top}_{t+1}; \mathbf{h}^{d\top}_{t+2}; \ldots; \mathbf{h}^{d\top}_{t+T_f}\right]^\top$.

4.1.2.2 Hierarchical temporal attention mechanism

Incorporating the information of similar days and hours has been considered in the literature for load forecasting; see e.g., [74, 75]. However, such information is often treated as additional input features or used to generate separate models. This work uses a novel hierarchical temporal attention layer designed from the previous chapter, which incorporates a similar day soft selection to re-evaluate the importance of historical information at each time step t.

Consider using previous M days of historical data to forecast the hourly loads for the next day, where each day includes $t_d = 24$ data points. Thus, we have $T_h = M \times t_d$ and $T_f = t_d$. Let $\mathbf{X}_i = (\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^n) \in \mathbb{R}^{t_d \times n}$ and $\mathbf{X}_f = (\mathbf{x}_f^1, \mathbf{x}_f^2, \ldots, \mathbf{x}_f^n) \in \mathbb{R}^{t_d \times n}$ collect the historical features for the day i and the future features of the next day. We use the sum of feature-by-feature dissimilarities $D(\mathbf{X}_i, \mathbf{X}_f) = \sum_{k=1}^n \|\mathbf{x}_i^k - \mathbf{x}_f^k\|_2$ to quantify the distance between all features of those two days. Then, the similar day weight γ_i is calculated as the softmax of the reciprocal of the distance:

$$\gamma_i = \frac{\exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)}{\sum_{i=1}^M \exp\left(D^{-1}(\mathbf{X}_i, \mathbf{X}_f)\right)},\ i = 1, 2, \ldots, M. \tag{4.5}$$

When forecasting load at time t, not all historical data contribute equally to the model's output. Hence, the attention mechanism facilitates the extraction of historical information that is more important to the current forecast value. Let subscript i denote the i-th day and j for j-th hour. Then, the attention weight $\beta_{i,j,t}$ is given by

$$\beta_{i,j,t} = \frac{\exp(d_{i,j,t})}{\sum_{i=1}^{M} \sum_{j=1}^{t_d} \exp(d_{i,j,t})}, \tag{4.6}$$

where $d_{i,j,t}$ is the $(i \times t_d + j)$-th element of vector $\mathbf{d}_t = (d_t^1, d_t^2, \ldots, d_t^{T_h})^\top \in \mathbb{R}^{T_h}$, which is given as

$$\mathbf{d}_t = \mathbf{V}_d \tanh\left(\mathbf{W}_d \left[{\mathbf{h}_{t-1}^d}^\top ; \mathbf{x}_t\right]^\top + \mathbf{b}_{wd}\right) + \mathbf{b}_{vd}. \tag{4.7}$$

The parameters $\mathbf{V}_d \in \mathbb{R}^{T_h \times d_h^{att}}$, $\mathbf{W}_d \in \mathbb{R}^{d_h^{att} \times (2hs+n)}$, $\mathbf{b}_{vd} \in \mathbb{R}^{T_h \times 1}$ and $\mathbf{b}_{wd} \in \mathbb{R}^{d_h^{att} \times 1}$ are trained jointly with the proposed model. $\mathbf{h}_{t-1}^d$ is the hidden vector of the decoder BiLSTM at time $t-1$.

To this end, let $\mathbf{h}_{i,j}$ denote the historical hidden state for the j-th hour in the i-th day from the encoder. The context vector of hierarchical temporal attention is calculated as $\mathbf{a}_t = \sum_{i=1}^{M} \sum_{j=1}^{t_d} \gamma_i \beta_{i,j,t} \mathbf{h}_{i,j}$.

4.1.3 Error Correction Module

Traditional error correction systems often involve creating a new model to forecast errors, resulting in higher learning costs and the potential loss of learnt knowledge obtained by the original predictive model. To overcome these shortcomings, a transfer-learning-based error correction module is proposed. Transfer learning utilizes previously acquired domain knowledge to solve new problems more efficiently yielding better results. In recent years, transfer learning has succeeded in supervised and unsupervised learning, including load forecasting, where knowledge is transferred between regions or

Algorithm 1: Training Procedure of the Framework

```
Input  : Forecasting module training set $\mathcal{D}_l = \{\mathbf{X}, \mathbf{y}_h, \mathbf{y}_f\}$,
         Error correction module training set
         $\mathcal{D}_e = \{\mathbf{X}_e, \mathbf{y}_{e,h}, \mathbf{y}_{e,f}\}$,
         Number of epochs $N_l$ and $N_e$,
         Feature weighting module $f_l$,
         Load forecasting module $g_l$
Output: Trained model $\{f_l, g_l, g_e\}$
Initialize model parameters;
for epoch = 1 to $N_l$ do
    for batch of $\{\mathbf{X}, \mathbf{y}_h, \mathbf{y}_f\} \in \mathcal{D}_l$ do
        $\tilde{\mathbf{X}} \leftarrow f_l(\mathbf{X})$;
        $\hat{\mathbf{y}}_f \leftarrow g_l(\tilde{\mathbf{X}}, \mathbf{y}_h)$;
        Compute training loss $L_{\text{LF}}$;
        Compute the gradient of loss;
        Update parameters in $g_l(\cdot)$ and $f_l(\cdot)$;
    end
end
$\hat{\mathbf{y}}_{e,f} \leftarrow g_l(f_l(\mathbf{X}_e), \mathbf{y}_{e,h})$;
$\mathbf{e} \leftarrow \mathbf{y}_{e,f} - \hat{\mathbf{y}}_{e,f}$;
Set feature weighting module $f_e(\cdot) = f_l(\cdot)$;
Set error correction module $g_e(\cdot) = g_l(\cdot)$;
Fixing the feature weighting layer, and train all other layers in the error
  correction module as follows:
for epoch = 1 to $N_e$ do
    for batch of $\{\mathbf{X}_e, \mathbf{e}_h, \mathbf{e}_f\}$ do
        $\tilde{\mathbf{X}} \leftarrow f_l(\mathbf{X})$;
        $\hat{\mathbf{e}}_f \leftarrow g_e(\tilde{\mathbf{X}}, \mathbf{e}_h)$;
        Compute training loss $L_{\text{EC}}$;
        Compute the gradient of loss;
        Update model parameter of $g_e(\cdot)$;
    end
end
Return trained model $\{f_l, g_l, g_e\}$.
```

households. However, transferring knowledge in error correction requires incorporating the error information into the target domain without changing the input dimension, while leveraging previously learned feature knowledge. The proposed error correction module addresses this challenge and aims to improve prediction accuracy by extracting valuable information from error values with the help of learned hidden features.

The error correction module is trained after the load forecasting module, which is first trained on dataset $\mathcal{D}_l$. Then, based on the error correction dataset $\mathcal{D}_e$, we compute the forecasting error as $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$, where $\mathbf{y}$ is the real output value and $\hat{\mathbf{y}}$ is the predicted value obtained by the forecasting model. The feature weighting module and error correction module are initialized by the forecasting model, with the feature weighting layer fixed and the other layers to be trained. To train the error correction module, a new dataset with feature input and forecasting error is generated and randomly split into training and validation sets. The algorithm computes the training loss and its gradient to update the error correction module via backpropagation. Upon completing the training of the error correction module, it can be used to correct forecast errors and improve forecasting accuracy. The final output is obtained as $\bar{\mathbf{y}} = \hat{\mathbf{y}} + \hat{\mathbf{e}}$, as shown in Fig. 4.1. The overall training procedure of the framework is summarized in Algorithm 1. The proposed transfer learning based model has several advantages including no need for hyper-parameter tuning and the ability to train with limited data. It also reuses existing knowledge learned by the original model, which results in a faster learning rate.

4.1.4 Loss Function

For the load forecasting module, we choose the mean squared error (MSE) loss and introduce ℓ_1 regularizer to encourage sparsity. The formulation is given as:

$$L_{\mathrm{LF}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda \left\|\boldsymbol{\alpha}\right\|_1, \tag{4.8}$$

where λ is the weighting parameter balancing the data fitting loss and the sparsity-promoting ℓ_1 penalty.

For the error correction module, we drop the ℓ_1 regularization term because the feature weighting layer is fixed. Hence, the loss function for training this module becomes

$$L_{\mathrm{EC}} = \frac{1}{N}\sum_{i=1}^{N}(e_i - \hat{e}_i)^2. \tag{4.9}$$

4.2 Experiment Setup

4.2.1 Data Description

The proposed framework is evaluated using two public datasets: the ISO New England (ISO-NE) dataset [1] and the North-American Utility (NAU) dataset[2]. ISO-NE annually releases reports that provide hourly historical demand and electricity pricing data for its control area and eight load zones. This work focuses solely on the control area dataset and ignores the price-related features. The input features include day-

[1]Available at https://www.iso-ne.com/isoexpress/web/reports/load-and-demand/-/tree/dmnd
[2]Available at https://class.ece.uw.edu/555/el-sharkawi/index.htm

ahead demand, dry bulb and dew point temperatures (in Fahrenheit), and time-related features. Data from 2015 to 2017 are used to train the forecasting module while 80% of 2018's data are randomly selected to train the error correction module. The remaining 20% data of 2018 are used for validation. The year 2019 is reserved for testing. The NAU dataset provides electricity load, temperature, and time information from January 1, 1985, to October 12, 1992. In our study, temperature and time-related features are considered. We used the data from 1987 to 1989 for training the forecasting model and randomly selected 80% of 1990 data to train the error correction module. The remaining 20% of 1990 is for validation, while the year 1991 is for testing.

4.2.2 Data Preparation

The missing values in the datasets are filled by linear interpolation. We incorporated time-related features such as indicators of weekends and holidays, seasons, hour of day, day of week, and month of year. We embedded categorical features via one-hot encoding [3] and standardize numerical features by subtracting their means and dividing by their standard deviations. The framework forecasts the next 24-hour' load demands using the previous seven days' load and features. The next 24 hours' features are assumed to be available as the model's input. For hourly data, we had $T_h = 168$ and $T_f = 24$. As shown in Fig. 4.4, the sliding window size is set to be 1 and 24 for training the forecasting module and error correction module, respectively. We list the model inputs from the ISO and NAU datasets in Tables 4.1 and 4.2.

[3]One-hot embedding represents each categorical variable as a binary vector with only one element is 1, while the rest are set to 0.

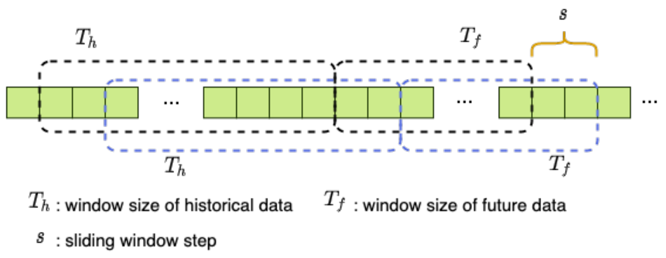

T_h
T_f
s
T_h
T_f
T_h : window size of historical data
T_f : window size of future data
s : sliding window step

Figure 4.4: Illustration of the sliding window step for data processing.

Table 4.1: Time-related features of the ISO-NE dataset.

Input	Size	Description
$\mathbf{y}_h$	168×1	Historical target values
DaDemd	192×1	Day ahead demand
DryBulb	192×1	Dry bulb temperature
DewPnt	192×1	Dew point temperature
Weekday	192×1	Weekday or weekend indicator
Holiday	192×1	Holiday or non-holiday indicator
Season	192×4	One-hot encoding
Hour of Day	192×24	One-hot encoding
Day of Week	192×7	One-hot encoding
Month of Year	192×12	One-hot encoding

Table 4.2: Time-related features of the NAU dataset.

Input	Size	Description
$\mathbf{y}_h$	168×1	Historical target values
Temperature	192×1	Historical and future temperature
Holiday	192×1	Holiday or non-holiday indicator
Season	192×4	One-hot encoding
Hour of Day	192×24	One-hot encoding
Day of Week	192×7	One-hot encoding
Month of Year	192×12	One-hot encoding

4.2.3 Baseline Models and Hyperparameters

To verify the effectiveness of our proposed framework, we compared four different types of models: classic machine learning models, DBN-based models, RNN-based models, and Transformer-based models. Details are given in below.

- Classic machine learning model: We tested SVR [12], RF [13], and GBM [15] using Scikit-Learn 0.23.2. Inputs to the model are historical and feature features and historical load that are flattened as a 1-D vector.
- DBN-based model: DBNs [81] are generative neural networks composed of multiple layers of RBMs. Each RBM layer is pre-trained in an unsupervised manner using the contrastive divergence algorithm, and the overall model is fine-tuned using supervised learning. Rough autoencoder combines rough set theory with DBNs, which can effectively handle uncertain and noisy data and learn complex patterns [82].
- RNN-based model: The CNN-LSTM [83] combines the advantages of both CNN and LSTM layers to improve forecasting accuracy. Attention-based load forecasting (ANLF) [80] is based on the encoder-decoder biLSTM architecture and utilizes a dynamic feature selection layer within the encoder. These models have shown promising results in load forecasting and can be used as effective baselines for future research in this field.
- Transformer-based model: Informer is a transformer-based model designed for time-series forecasting as proposed in the 2021 AAAI Best Paper [84]. Unlike the

RNN-based model, transformers can handle sequential data in parallel to reduce training time.

The computing environment is a machine with 3.7 GHz Intel Core i7-8700K Six-Core and NVIDIA GeForce GTX 1080 Ti (11GB GDDR5X). The training time for each epoch is around seven minutes for the proposed model. Deep learning based models are trained by using Adam optimizer and implemented with PyTorch 1.6.0. The initial learning rate is 0.001 which decays by 0.1 times for every 30 epochs. Early stopping criteria is set with patience 30. All models share the same training, validation, and testing data samples and input features for fair comparisons. We performed a grid search to identify the best hyper-parameter set based on the validation data. Grid search is commonly used for hyper-parameter tuning, which involves setting a range of values for each hyper-parameter and testing all possible combinations. The details of the grid search are given in Table 4.3.

4.2.4 Performance Metrics

The mean absolute error (MAE) and mean absolute percentage error (MAPE) are used to evaluate the forecasting accuracy. They are defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \left| y_i - \hat{y}_i \right|, \tag{4.10a}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \tag{4.10b}$$

Table 4.3: The ranges of hyper-parameter tuning. Bold and italic fonts indicate the best values for the ISO-NE and the NAU datasets, respectively.

Model	Hyper-parameter range
SVR [12]	Kernel = (*RBF*, Linear, **Poly**)
	Degree = (2, 3) with Poly
	Gamma = (*auto*, **scale**) with Poly/RBF
	C = (0.1, 1, ***10***)
RF [13]	Number of estimators = (100, *500*, **1000**)
	Maximum depth = (5, 10, ***20*** None)
	Minimum samples split = (***2***, 5)
	Minimum samples leaf = (***1***, 3, 5, 10)
GBM [15]	Loss = (***ls***, lad, huber, quantile)
	Learning rate = (*0.1*, **0.01**)
	Number of estimators = (100, 500, ***1000***)
	Maximum depth = (***5***, 10, 20, None)
	Minimum samples split = (*2*, **5**)
	Minimum samples leaf = (1, 3, *5*, **10**)
DBN [81]	Hidden layer 1 = (*128*, **256**, 512)
	Hidden layer 2 = (*128*, **256**, 512)
	Hidden layer 3 = (***0***, 64, 128)
	Batch = (***64***, 128)
RAE [82]	Hidden layer 1 = (128, ***256***, 512)
	Hidden layer 2 = (128, ***256***, 512)
	Hidden layer 3 = (***0***, 64, 128)
	Batch = (64, ***128***)
CNN-LSTM [83]	Batch = (**64**, *128*)
	Hidden size = (128, 256, ***512***)
	Kernal size = (*3*, 5, **8**)
ANLF[80]	Batch = (64, ***128***)
	Hidden size = (***128***, 256, 512)
Informer[84]	Batch = (***64***, 128)
	Hidden size = (**64**, *128*, 256, 512)
	label length = (0, 24, ***48***)
	number of attention heads = (2, 4, ***8***)
PM-LSTM	Batch = (*64*, **128**)
	Hidden size = (**128**, *256*, 512)
	λ = (0, ***0.001***, 0.01)
PM-GRU	Batch = (64, ***128***)
	Hidden size = (*128*, 256, **512**)
	λ = (***0***, 0.001, 0.01)

where y_i and $\hat{y}_i$ are the i-th true and predicted outputs. n is the number of points in the testing horizon.

4.3 Simulation Results

In this section, three case studies are carried out to show the effectiveness of the proposed framework. Case 1 shows the ablation study results[4]. Case 2 compares the baseline models in section III-C and our proposed model. Case 3 shows the generalization capability, for which we added the feature weighting mechanism and error correction module to the Informer.

4.3.1 Case 1: Ablation Study and Discussion

An ablation study is conducted based on the NAU dataset. Table 4.4 presents the MAE and MAPE results. The first row shows the performance of the backbone encoder-decoder based BiLSTM model. We then compared three different approaches of feature weighting: mutual information (MI) [85], random forest (RF) [26], and our proposed feature weighting attention (FW). In addition, we evaluated the performance of the backbone model with two types of temporal attention mechanisms: single layer temporal attention (TA) and the proposed hierarchical temporal attention, which incorporates similar day information (SDA). Finally, we included the results for two error correction methods: the baseline ARIMA model (BL) [28] and our proposed feature-

[4]Ablation study is a method of understanding the contribution of different components of a system to its overall performance.

reinforced error correction model (EC). The results show that the proposed feature weighting and error correction outperform the existing methods. Each individual module improves the accuracy of the backbone model. Moreover, the combination of these modules further enhances the performance. Compared with the other competing alternatives, our proposed framework achieves a significant improvement in accuracy.

The visualization of the performance improvements is shown in Fig 4.5. First, the proposed weighting attention can identify feature importance in the time domain. Fig 4.5(c) shows that our method adds time-varying weights on different features while mutual information approach exerts time-invariant weights shown in Fig 4.5(a). Second, our method puts more accurate weights on each feature compared with RF. Our weight assignments are sparser, with higher weights on temperature and hour [cf. Fig. 4.5(c)]. In contrast, RF yields similar feature weights at approximately 0.02 [cf. Fig. 4.5(b)]. Third, our approach shows a good response to the input changes while RF is ignorant of different input data. Finally, the clear pattern of feature weights in Fig. 4.5(c) indicates that the temperature from 9 AM to 4 PM is a more critical factor, and the hour is not as important; however, for the rest of the time, hour information also contributes to the forecasting value.

4.3.2 Case 2: Load Forecasting Model Comparison

Besides our proposed model (FW+TA+SDA in Table 4.4) with LSTM (PM-LSTM) and GRU (PM-GRU) implementations, we tested eight baseline models. To have a fair comparison, the error correction module was deactivated because it is not

Table 4.4: NAU dataset: Ablation study for the proposed framework. Acronyms: MI (mutual information feature weight), RF (random forest feature weight), FW (feature weighting attention), TA (temporal attention), SDA (similar day attention), BL (baseline ARIMA error correction) and EC (the proposed error correction).

MI [85]	RF [26]	FW	TA	SDA	BL [28]	EC	MAE	MAPE (%)
-	-	-	-	-	-	-	89.20	3.93
✓	-	-	-	-	-	-	59.93	2.57
-	✓	-	-	-	-	-	63.65	2.76
-	-	✓	-	-	-	-	56.65	2.45
-	-	-	✓	-	-	-	69.10	2.96
-	-	-	✓	✓	-	-	64.29	2.78
-	-	✓	✓	✓	-	-	48.96	2.15
-	-	✓	✓	✓	✓	-	46.36	2.09
-	-	✓	✓	✓	-	✓	**45.70**	**2.00**

Table 4.5: Forecasting errors over the year 2019 for the ISO-NE dataset.

Model	SVR [12]	RF [13]	GBM [15]	DBN[81]	RAE[82]
MAE	317.49	393.58	277.54	326.45	308.36
MAPE (%)	2.33	2.87	2.00	2.38	2.22
Model	CNN-LSTM [83]	ANLF[80]	Informer[84]	PM-LSTM	PM-GRU
MAE	309.06	258.70	256.89	**229.47**	**231.84**
MAPE (%)	2.27	1.88	1.89	**1.66**	**1.67**

Table 4.6: Forecasting errors over the year 1991 for the NAU dataset.

Model	SVR [12]	RF [13]	GBM [15]	DBN[81]	RAE[82]
MAE	67.97	99.91	83.05	68.33	63.55
MAPE (%)	2.98	4.40	3.61	2.99	2.79
Model	CNN-LSTM [83]	ANLF[80]	Informer[84]	PM-LSTM	PM-GRU
MAE	61.85	58.65	57.13	**48.96**	**46.42**
MAPE (%)	2.73	2.58	2.49	**2.15**	**2.03**

directly applicable to those classic machine learning algorithms such as SVR, RF and GBM. Table 4.5 and Table 4.6 show the forecasting error for the two datasets.

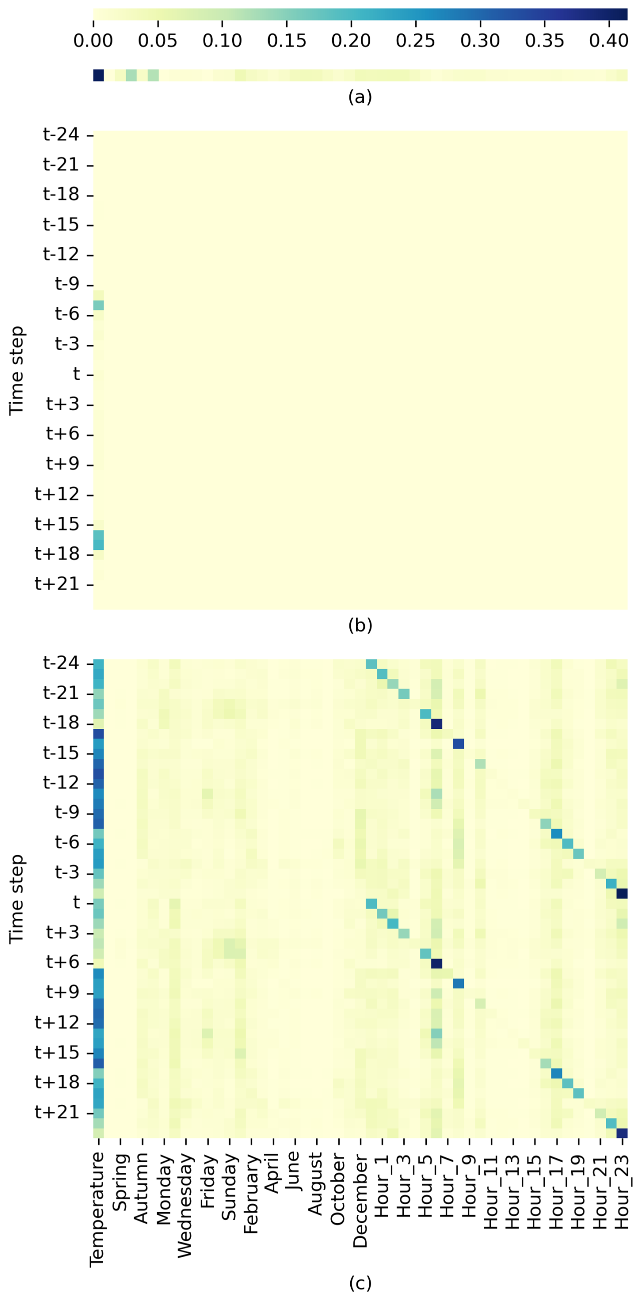


Figure 4.5: Two-day feature weight visualization for the NAU dataset with different approaches: (a) mutual information, (b) random forest, and (c) the proposed feature weighting attention.

Among the three classic machine learning methods, GBM performs the best for the ISO-NE dataset, while SVR stands out for the NAU dataset. DBN-based models attain results that are comparable to CNN-LSTM for both datasets. Notably, our proposed models PM-LSTM/GRU consistently outperform all the other models for both datasets with the smallest forecasting errors. Moreover, comparing the ANLF model with the proposed ones, extracting the feature weighting layer further improves the accuracy and increases the model's generalization capability. Overall, these findings highlight the effectiveness of our proposed approach. The detailed forecasting performance over three days are given in Fig. 4.6 and Fig. 4.7, where the relative error (RE) between the forecast value $\hat{y}$ and the true value y is defined as

$$\mathrm{RE} = \frac{|y - \hat{y}|}{y} \times 100\%. \tag{4.11}$$

4.3.3 Case 3: Generalization Capability

To further show the generalization capability of the proposed framework, we applied both the feature weighting and error correction to the transformer-based Informer. The result is reported in Table 4.7 for the ISO-NE dataset. We compared the model itself with the model having feature weighting and/or error correction. The forecasting curves and relative errors are shown in Fig. 4.8. By the ablation study, the model with our proposed feature weighting and error correction mechanisms performs the best. This verifies the merit of integrating the feature weighting to provide more

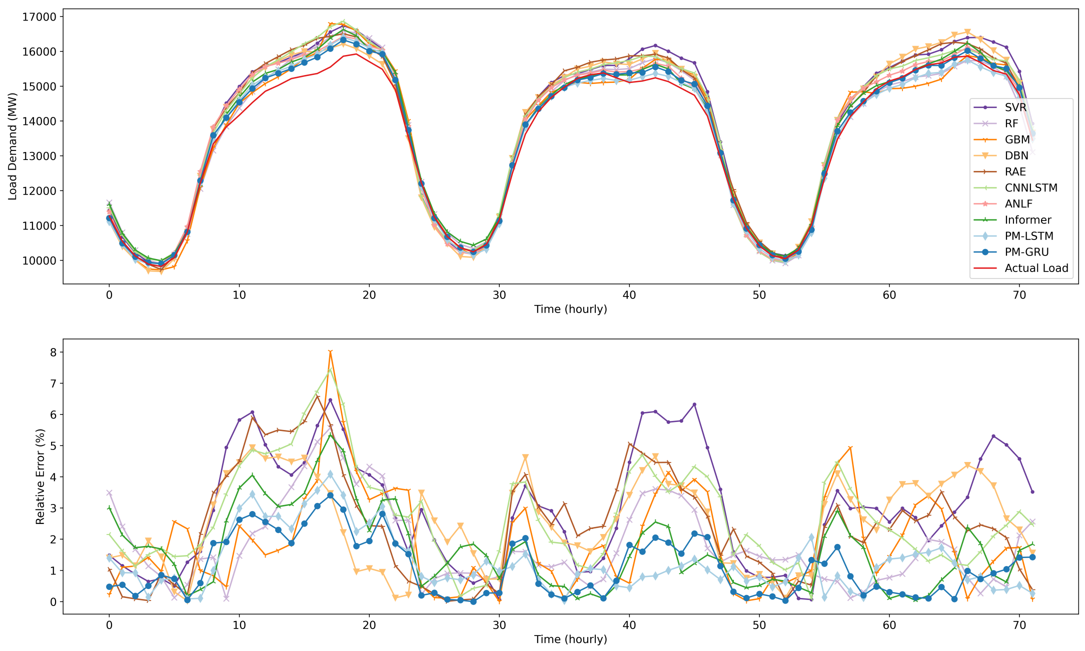


Figure 4.6: Forecasting curve and relative error for the ISO-NE dataset (3 days).

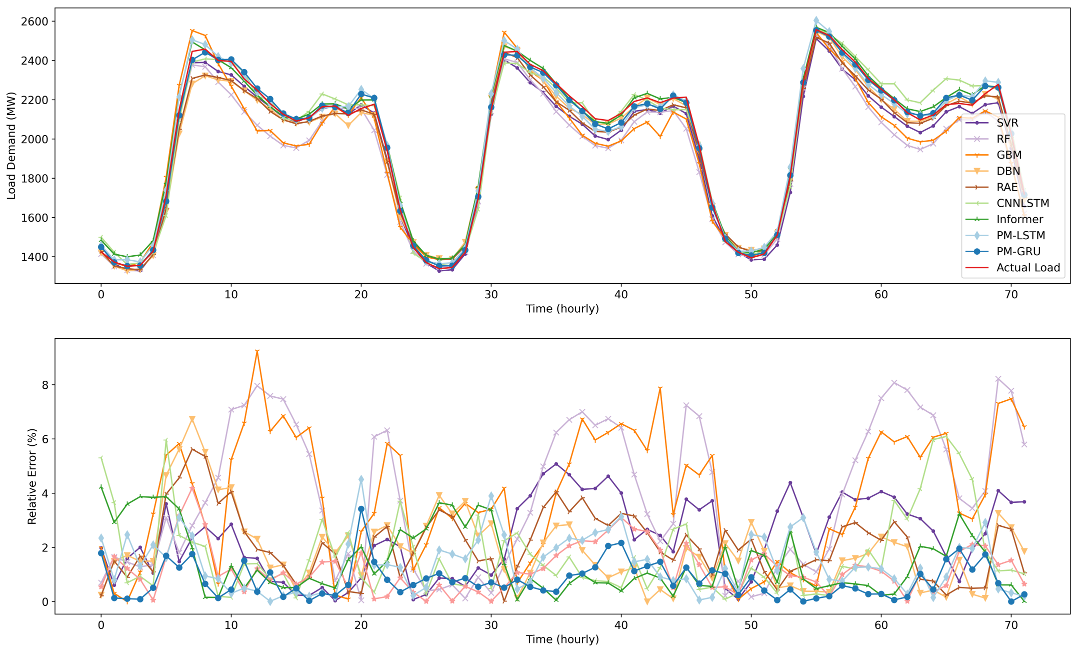


Figure 4.7: Forecasting curve and relative error for the NAU dataset (3 days).

Table 4.7: ISO-NE dataset: Ablation study for Informer with feature reinforced error correction (EC) and/or feature weighting attention (FW).

FW	EC	MAE	MAPE (%)
-	-	256.89	1.89
✓	-	249.17	1.81
-	✓	239.35	1.76
✓	✓	**239.23**	**1.74**

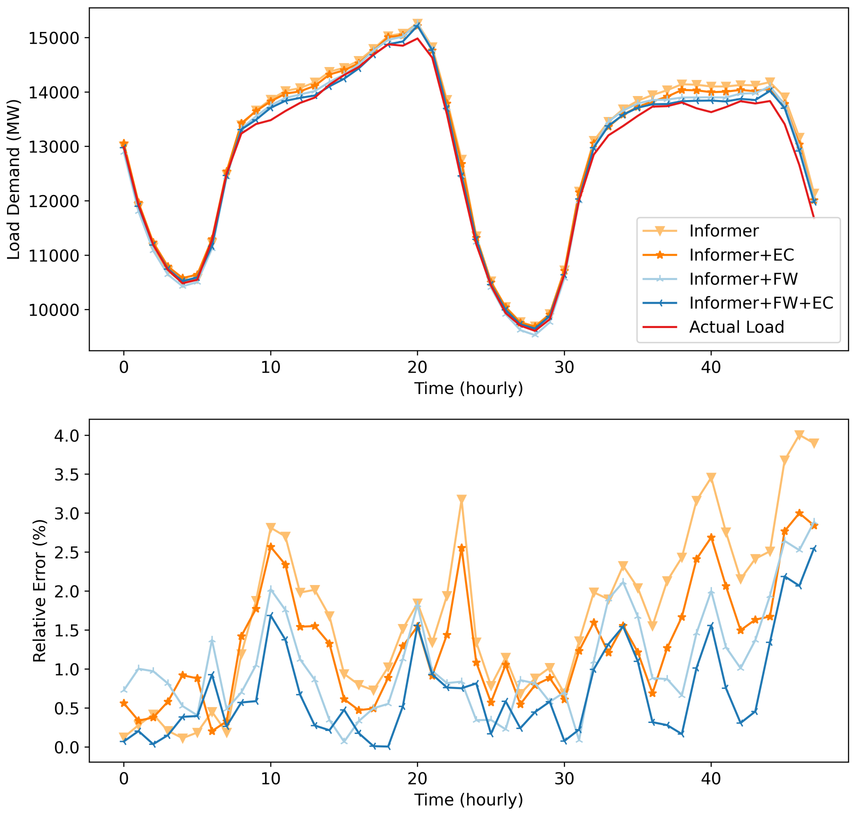


Figure 4.8: ISO-NE dataset: The forecasting curves and relative errors for Informer, Informer with error correction, Informer with feature weighting, and Informer with both feature weighting and error correction (for two days).

informative features and error correction to further improve the accuracy.

4.4 Summary

This chapter develops a unifying deep learning framework for multi-horizon STLF. Three interactive modules are developed with high generalization capability, which includes the feature weighting mechanism, STLF model and error correction module. In the proposed framework, the feature weighting mechanism is designed to provide informative input features for both historical and future time horizons. The STLF model with a hierarchical temporal attention layer decodes the next-day load with the future input features and similar temporal information. The hierarchical temporal attention layer provides a natural way to incorporate similar day information. In addition, the error correction module is developed based on transfer learning. It can reuse the learned hidden feature extraction to reduce the training cost. The modular design of our framework facilitates customization and independent modification. The extensive simulation results tested on the two datasets corroborate the merits of our framework.

Chapter 5

MATNilm: Multi-appliance-task Non-Intrusive Load Monitoring with Limited Labeled Data

Non-intrusive load monitoring (NILM) identifies the status and power consumption of various household appliances by disaggregating the total power usage signal of an entire house. Efficient and accurate load monitoring facilitates user profile establishment, intelligent household energy management, and peak load shifting. This is beneficial for both the end-users and utilities by improving the overall efficiency of a power distribution network. Existing approaches mainly focus on developing an individual model for each appliance. Those approaches typically rely on a large amount of household-labeled data which is hard to collect. In this chapter, a multi-appliance-task framework is proposed with a training-efficient sample augmentation (SA) scheme that boosts the disaggregation performance with limited labeled data. For each appliance, a shared-hierarchical-split structure is developed for its regression and classification tasks. In addition, we also propose a two-dimensional attention mechanism in order to capture spatio-temporal correlations among all appliances.

5.1 Preliminary

Machine learning models for NILM aim to learn a mapping from aggregated power consumption (the input signal) to individual appliances' power consumption (the output signal). In order to train a model, a household labeled set of input (aggregate power consumption) - output (appliances power consumption) pairs $\{(x_i, \mathbf{y}_i)\}_{i=1}^{T_l}$ are used where T_l is the number of time steps. Typically, a sliding window method is used to construct the training dataset

$$\mathcal{D} = \left\{ (\mathbf{x}_{t:t+T-1}, \mathbf{y}_{t+w:t+T-w-1}) \mid t \in \{s * k\}_{k=1}^{N} \right\},$$

where T is the input length, s is step size, N is the number of training examples and w is the additional window size. For context-aware setup [49][52][54], we have $w > 0$, which means the length of the input is longer than the length of the output, providing 'extra' input context to output.

Let $\mathbf{x}_{t:t+T-1} = (x_t, x_{t+1}, \dots, x_{t+T-1})^\top \in \mathbb{R}^T$ denotes the aggregate household load consumption, $\mathbf{y}^i_{t:t+T-1} = (y^i_t, y^i_{t+1}, \dots, y^i_{t+T-1})^\top \in \mathbb{R}^T$ denotes the corresponding power consumption of appliance i to be disaggregated in the same period, the remaining unlabeled appliance consumption as $\mathbf{u}_{t:t+T-1} = (u_t, u_{t+1}, \ldots, u_{t+T-1})^\top \in \mathbb{R}^T$. Thus, the aggregate load consumption at each time step t would be the summation of appliance load consumption and unknown appliance load consumption:

$$x_t = \sum_{i \in \mathcal{K}} y_t^i + u_t + \epsilon_t, \tag{5.1}$$

where ϵ is the measurement noise, $\mathcal{K}$ is the set of appliances to be disaggregated.

For the NILM task, given the input as the aggregate load consumption $\mathbf{x}$, we aim to separate the signal to each appliance $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^{|\mathcal{K}|}$. The majority of existing methods develop $|\mathcal{K}|$ independent models to learn each appliance's mapping:

$$\hat{\mathbf{y}}^i_{t+w:t+T-w-1} = f^i(\mathbf{x}_{t:t+T-1}). \tag{5.2}$$

We define a multi-appliance-task training schema

$$\{\hat{\mathbf{y}}^1_{t+w:t+T-w-1}, \hat{\mathbf{y}}^2_{t+w:t+T-w-1}, \ldots, \hat{\mathbf{y}}^{|\mathcal{K}|}_{t+w:t+T-w-1}\} = f(\mathbf{x}_{t:t+T-1}), \tag{5.3}$$

where f is trained jointly by all appliances in set $\mathcal{K}$.

5.2 Methodology

In this section, we present our proposed framework for NILM with limited household labeled data. We first introduce a novel sample augmentation scheme, which is followed by the MAT structure with 2DMA.

5.2.1 Sample Augmentation (SA) Algorithm

A deep learning model typically requires a substantial amount of training data in order to achieve good generalization capability. This “massiveness” refers not only to the quantity of training samples, but also to the diversity of the data. Existing approaches rely on numerous household labeled data $\{(x_i, \mathbf{y}_i)\}_{i=1}^{T_l}$. However, collecting labeled data is not a simple task and often involves the installation of specialized hardware, which can be time-consuming and raise privacy concerns. Moreover, the collected dataset may suffer from various issues, including time inconsistencies, missing data, and data of poor quality. Additionally, due to the fact that some appliances are predominantly in the OFF state, the dataset can be highly imbalanced. Lastly, the available labeled samples are often limited and do not adequately represent the wide range of appliance models and usage patterns.

To address the aforementioned challenges, we propose a dynamic sample augmentation algorithm to generate appliance profiles based on real data. As shown in (5.1), aggregate data can be synthetically generated by the combination of the appliance's operation profiles. An appliance's operation profile is defined as a sequence of sampled power measurements over one complete operation cycle [64]. Based on this observation, we can generate synthetic labeled data $\{(x_i, \mathbf{y}_i)\}_{i=1}^{T_l}$ according to comprehensive operation profiles, which are assumed to be available. The proposed algorithm augments the training sample adaptively until new instances cannot improve the performance further.

Our algorithm is based on a newly constructed appliance pool $\mathcal{A} = \cup_{i=1}^{N_a} \mathcal{A}_i$, which collects power consumption signals of N_a appliances; see also [68]. For appliance i, $\mathcal{A}_i$ is the set N_i operation profiles $\{\tilde{\mathbf{y}}_j^i\}_{j=1}^{N_i}$. The appliance pool can be built by i) real operation profiles where measurements are provided by existing datasets, appliance companies, or smart plugs; and ii) synthetic operation profiles produced by generative models such as GAN, diffusion models, or other algorithms.

To further diversify the training data, we utilize time series augmentation methods to modify existing profiles via vertical scale, horizontal scale, and mixed vertical-horizontal scale. In vertical scaling, the signal magnitude of $\mathbf{y}^i$ is scaled by α times, i.e., $\mathbf{y}^i \times \alpha$, where $\alpha \sim N(1, \sigma^2)$ is a Gaussian random variable with mean one and variance σ^2. In horizontal scaling, the signal length l of $\mathbf{y}^i$ is scaled (compressed or extended) by β times, i.e., $l \times \beta$, where $\beta \sim N(1, \sigma^2)$. Linear interpolation is used to generate unknown data points while scaling. Other modification methods could be added accordingly.

Let prob^i denote the probability that the profile of appliance i gets sample augmentation. If needed, we can tune this parameter to balance the on/off sampling rate. In addition, define $p^i = [p^i_{\text{IN}}, p^i_{\text{V}}, p^i_{\text{H}}, p^i_{\text{M}}]$ as the probabilities of four modes: intact, vertical scaling, horizontal scaling and mixed scaling. The SA algorithm is applied once a batch of samples is selected during training. As the training continues, new training samples will be continuously generated, which may improve the generalization capability of a machine learning model. The training will stop once the new training instance cannot contribute to the performance improvement, or the maximum training epoch has been reached. The detailed implementation of the proposed sample augmentation is given in Algorithm 2.

Algorithm 2: Sample Augmentation Algorithm

```
Input  : Original mini-batch $\bar{\mathcal{D}}$ from training set $\mathcal{D}$;
         Pre-defined sample augmentation probability for each appliance
         $\text{prob}^i$;
         Pre-defined modify-mode probability mass function for each
         appliance $p^i$; and
         Appliance pool $\mathcal{A}$
Output: Augmented mini-batch $\bar{\mathcal{D}}_a$
for $(\mathbf{x}, \mathbf{y}, \mathbf{y}_c) \in \bar{\mathcal{D}}$ do
    for appliance $i$ in $\mathcal{A}$ do
        $r \leftarrow$ generate a random number from a uniform distribution over
          $[0, 1]$;
        if $r < \text{prob}^i$ then
            $sig \leftarrow$ randomly select an operation profile $\tilde{\mathbf{y}}^i_j$ from $\mathcal{A}_i$;
            $mode \leftarrow$ randomly choose a modification mode based on the
              distribution $p^i$;
            $sig\_new \leftarrow$ modify $sig$ according to the chosen mode, and
              randomly slice $sig\_new$ to the same length of $\mathbf{y}^i_j$;
            if $i \in \mathcal{K}$ then
                $\mathbf{X} \leftarrow \mathbf{X} - \mathbf{y}^i_j$;
                $\mathbf{y}^i \leftarrow sig\_new$;
                $\mathbf{y}^i_c \leftarrow$ update on/off status;
            end
            $\mathbf{X} \leftarrow \mathbf{X} + sig\_new$;
        end
    end
end
```

Remark 1 (Merits of sample augmentation (SA))**.** *The proposed SA offers several advantages over existing data augmentation methods that generate a synthetic dataset. First, it does not require a predefined size of the augmented dataset. SA with early stopping criteria enables an efficient generation of diverse training data until no further performance improvements. This property is not present in existing augmentation methods that often rely on ad-hoc or trial-and-error ways to determine the amount of* *needed data samples. Second, performing SA within the training process enables end-to-end training of the model. This simplifies the training pipeline and eliminates the need for a separate data preprocessing step. It makes the training more efficient with reduced memory overhead. Compared with existing DA schemes that require generating and storing a separate augmented dataset, end-to-end SA generates augmented data on the fly during the training. This further enables the model to learn from a diverse range of augmented data and avoid overfitting. Finally, in our scenario of limited data, existing DA approaches need to generate a synthetic dataset by creating multiple copies of the original dataset, based on which augmentation techniques can be applied. This approach is computationally expensive and time-consuming. In contrast, the proposed SA does not require an increase in the length of the dataset, which is more computationally efficient.*

5.2.2 Multi-appliance-task Network Architecture

Multi-task learning can improve a model's generalization capability and performance, especially when the tasks share similar knowledge that can be leveraged by one another. In the context of NILM tasks, the model acts as a resource allocator that distributes aggregate power into each appliance. This means that the power assigned to one appliance depends on the power assigned to the others, as the sum of each disaggregated value cannot exceed the aggregate input inherently. To model this connection, we designed a MAT network with 2DMA. The backbone of the network consists of an encoder and decoder structure, where the encoder learns a shared representation for all appliances, and the decoder tries to distribute power into each appliance. The decoder is split into n branches, with each branch further split into two branches for regression and classification tasks. We stacked m decoder blocks in the decoder to extract higher-level features from the shared representation, with each decoder block composed of a 2DMA layer and a fully connected feed-forward layer. We also employed residual connection and layer normalization for each sub-layer, following the design from the transformer. The overall structure is shown in the Fig.5.1.

5.2.2.1 2DMA mechanism

The 2DMA mechanism comprises two attention layers: temporal attention and appliance-wise attention. The temporal attention layer is specifically designed to capture the temporal correlations across time for a given appliance by selectively attending to different time steps within the input sequence. This layer enables the model to effectively capture temporal dependencies. On the other hand, the appliance-wise attention layer is implemented to exploit the representations of other appliances at the same time step.

The 2DMA architecture is built based on the multi-head (MH) attention, which is a powerful mechanism for exploring complex relationships among input elements.

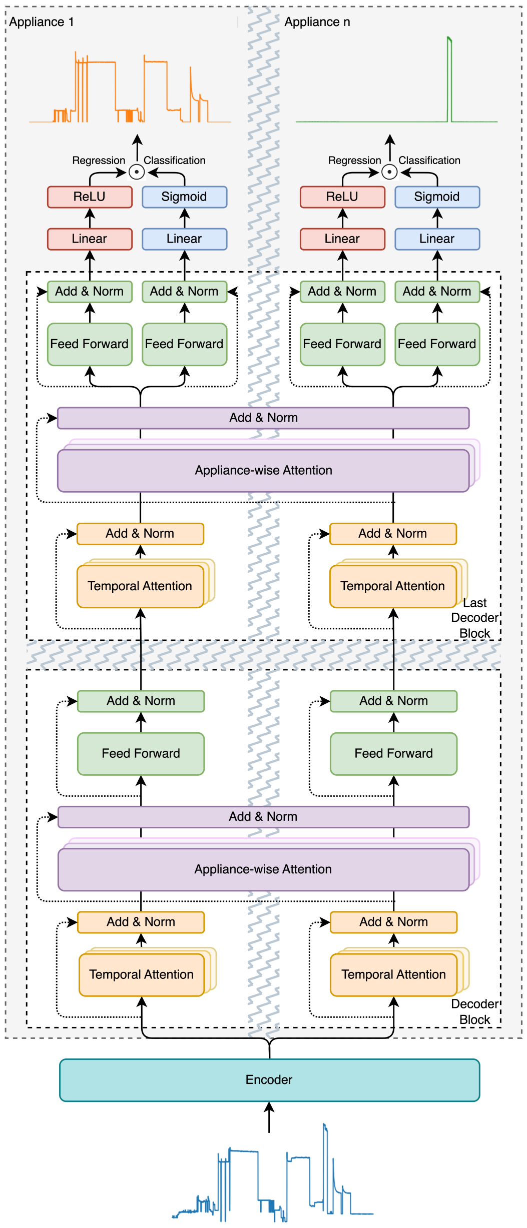


Figure 5.1: Multi-appliance-task network architecture with temporal attention and appliance-wise attention. The decoder comprises a stack of m identical decoder blocks with n branches. m and n are the numbers of decoder blocks and the number of appliances to be disaggregated, respectively. This figure only shows the case when $n = 2$ and $m = 1$. The grey zigzag areas can be extended as n and m increase.

Specifically, multi-head attention is defined as [22]:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{5.4a}$$

$$\text{head}_\text{i} = \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right), \tag{5.4b}$$

$$\text{MH}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right)\mathbf{W}^O. \tag{5.4c}$$

Equation (5.4a) is the scaled dot-product attention that takes the input in the form of three matrices, known as the query $\mathbf{Q}$, the key $\mathbf{K}$, and the value $\mathbf{V}$. Essentially, the attention function maps each query and a set of key-value pairs to an output, which is computed as a weighted sum of the values. The weight assigned to each value is given by a similarity score between the query and the corresponding key. Then, we perform the attention computation in parallel with different projected versions of queries, keys and values to produce multiple attention heads in (5.4b). To this end, those independent attention heads are concatenated and transformed with another learned linear projection $\mathbf{W}^O$ to obtain the final output (5.4c). Such a linear projection, which serves to reduce the dimension of the long concatenated head, learns which attention heads to take more notice of. All above matrices $\mathbf{W}$ are learnable parameters obtained by e.g., backpropagation.

The self-attention learns how best to route information between pieces (known as tokens) of an input sequence while the multi-head version enables the model to jointly attend to information from different representation subspaces at different positions [22]. This ubiquitous mechanism facilitates the creation of richer and better representations that can boost performance on various machine learning tasks.

In the temporal attention, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ come from the hidden representations of the same appliance learned from the previous layer $\mathbf{h}^i_{t+w:t+T-w-1}$ and are updated across different time steps:

$$\boldsymbol{\theta}^i_{t+w:t+T-w-1} = \text{LayerNorm}\left(\mathbf{h}^i_{t+w:t+T-w-1} + \text{MH}\left(\mathbf{h}^i_{t+w:t+T-w-1}\right)\right). \tag{5.5}$$

Similarly, in the appliance-wise attention, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ come from the hidden representations of each appliance at the same time step $\boldsymbol{\theta}^{i\in\mathcal{K}}_t$. Each appliance can update its representation based on other appliances' representations:

$$\boldsymbol{\phi}^{i\in\mathcal{K}}_t = \text{LayerNorm}\left(\boldsymbol{\theta}^{i\in\mathcal{K}}_t + \text{MH}\left(\boldsymbol{\theta}^{i\in\mathcal{K}}_t\right)\right). \tag{5.6}$$

5.2.2.2 MATNilm architecture

Given an aggregated power consumption signal $\mathbf{x}_{t:t+T-1}$, the encoder learns a hidden shared representation for better separating by later appliance-specific split task

$$\mathbf{h}_{t+w:t+T-w-1} = \text{Encoder}\left(\mathbf{x}_{t:t+T-1}\right), \tag{5.7}$$

where $w = 0$ is for general setup and $w > 0$ is for context-aware setup. The decoder splits a branch for each appliance $i \in \mathcal{K}$ with initial input $\mathbf{h}_{t+w:t+T-w-1}$. Each decoder block then updates the hidden representation via the 2DMA layer

$$\boldsymbol{\phi}_t^i = \text{2DMA}\left(\mathbf{h}_{t+w:t+T-w-1}\right). \tag{5.8}$$

Then, the representation of each appliance is updated by a fully connected feed-forward network

$$\mathbf{h}_t^i = \text{LayerNorm}\left(\boldsymbol{\phi}_t^i + \max\left(0, \boldsymbol{\phi}_t^i \mathbf{W}_1^i + \mathbf{b}_1\right)\mathbf{W}_2^i + \mathbf{b}_2\right). \tag{5.9}$$

In the last decoder block, we further split the network for each appliance in order to explicitly exploit power consumption and on/off state [52]:

$$\mathbf{h}_{c,t}^i = \text{LayerNorm}\left(\boldsymbol{\phi}_t^i + \max\left(0, \boldsymbol{\phi}_t^i \mathbf{W}_{c,1}^i + \mathbf{b}_{c,1}\right)\mathbf{W}_{c,2}^i + \mathbf{b}_{c,2}\right), \tag{5.10a}$$

$$\mathbf{h}_{r,t}^i = \text{LayerNorm}\left(\boldsymbol{\phi}_t^i + \max\left(0, \boldsymbol{\phi}_t^i \mathbf{W}_{r,1}^i + \mathbf{b}_{r,1}\right)\mathbf{W}_{r,2}^i + \mathbf{b}_{r,2}\right). \tag{5.10b}$$

In the regression subnetwork, a fully connected layer with the rectified linear unit (ReLU) activation function is used to transform the hidden information to the estimated power consumption for i-th appliances at each time step:

$$\hat{p}_t^i = \text{ReLU}(\mathbf{h}_{r,t}^i \mathbf{W}_r^i + \mathbf{b}_{wr})\mathbf{V}_r^i + \mathbf{b}_{vr}. \tag{5.11}$$

Similarly, in the classification subnetwork, a fully connected layer with the sigmoid activation function is used to transform the hidden information to the estimated on/off status for i-th appliances at each time step:

$$\hat{o}_t^i = \text{Sigmoid}(\mathbf{h}_{c,t}^i \mathbf{W}_c^i + \mathbf{b}_{wc})\mathbf{V}_c^i + \mathbf{b}_{vc}. \tag{5.12}$$

The final estimated power consumption for appliance i is calculated by $\hat{y}_t^i = \hat{p}_t^i \times \hat{o}_t^i$.

5.2.3 Loss Function

In this work, we extend the loss function in [52] to satisfy the MAT setup, which is defined given as:

$$\mathcal{L} = \sum_{i=1}^{|\mathcal{K}|} (\mathcal{L}_{\text{output}}^i + \mathcal{L}_{\text{on}}^i), \tag{5.13}$$

where $\mathcal{L}_{\text{output}}$ is the mean squared error (MSE) of the network final output:

$$\mathcal{L}_{\text{output}}^i = \frac{1}{T} \sum_{t}^{T} \left(y_t^i - \hat{p}_t^i \hat{o}_t^i \right)^2, \tag{5.14}$$

and $\mathcal{L}_{\text{on}}$ is the binary cross entropy (BCE) of the classification subnetwork's result:

$$\mathcal{L}_{\text{on}}^i = -\frac{1}{T} \sum_{t=1}^{T} \left(o_t^i \log \hat{o}_t^i + \left(1 - o_t^i\right) \log \left(1 - \hat{o}_t^i\right) \right). \tag{5.15}$$

5.3 Experiment Setup

5.3.1 Data Preprocessing

The proposed algorithms are tested on two real-world datasets: Reference Energy Disaggregation Data Set (REDD) [86] and UK-DALE [87]. The REDD dataset comprises 6 US household aggregate load consumption data with time ranges varying from 23 to 48 days. The time resolution of aggregate signal is 1 second while 3 seconds for appliance-wise signals. To ensure the consistency of time alignment, the aggregate signal is downsampled to 3 seconds to match the appliance-wise sampling rate. On the other hand, the UK-DALE dataset includes 5 UK households, with house 1 having more than four years of recordings. The time resolution for aggregate signal and appliance-wise signals is 6 seconds. The appliances of interest include fridge, dishwasher, microwave and washer dryer for REDD while an additional appliance kettle is being considered for UK-DALE.

We followed the data preprocessing procedure in [52]. In that study, each pair of the aggregate-appliance signal is processed independently, resulting in potentially different timestamps for the processed appliance sequences. However, for the multi-appliance-task framework, it is crucial to have a correct time alignment between each appliance signal and its corresponding aggregate one. To ensure that, we first utilized the merged dataset given by NILMTK [88], where the aggregated and individual appliance active power consumptions are merged into a single table based on the timestamps. Next, we split the table to exclude timestamps with 20 continuous missing values or

1200 continuously unchanged values for any signals. We then used the backward-filling method to fill in the remaining missing values. We only considered sub-tables with more than one hour of duration to ensure a sufficient amount of data for further analysis. The entire dataset is normalized (divided by 612) in the same fashion as suggested by [52].

5.3.2 Three Scenarios

We considered three scenarios in this section: Scenario one (S1) serves as a baseline when the entire dataset is available, following the common practice of prior works. For the REDD dataset under S1, houses 2-6 are used for training while house 1 is for testing; see also [52] and [54]. We selected parts of data from houses 2 and 3 for validation because they have the active state for all the appliances to be estimated. Scenario two (S2) is designed to evaluate the proposed method with limited labeled data. We considered this scenario because obtaining long-term household-level labeled data is often challenging in practice. Specifically, we considered an unfavorable scenario where only a single day's worth of labeled data is accessible for training, while another day's worth of data is utilized for validation and testing on an unseen household. Scenario three (S3) is built upon S2 by incorporating limited appliance activation measurements to augment the training samples. Table 5.1 presents an overview of the training, validation, and testing datasets used in this study.

Table 5.1: Training, validation, and testing datasets for REDD and UK-DALE.

		House #	Time index
REDD	Training	3	2011-04-21 19:41:24 - 2011-04-22 19:41:21
	Validation	3	2011-05-23 10:31:24 - 2011-05-24 10:31:21
	Testing	1	2011-04-18 09:22:12 - 2011-05-23 09:21:51
UK-DALE	Training	1	2017-04-23
	Validation	1	2017-04-25
	Testing	2	2017-04-12 - 2017-04-25

5.3.3 Model Details

The SGN-Conv model is implemented based on the specifications in [52]. In addition, we also explored a variant of SGN called SGN-LSTM, where each subnetwork is composed of five layers of BiLSTM followed by two linear layers with a hidden size of 32. The MAT-Conv model uses the convolutional layers of SGN-Conv as its encoder, while the LSTM layers in SGN-LSTM serve as the encoder for MAT-LSTM. For each target appliance, three decoder blocks are employed in this study.

For REDD, the length of the input sequence is fixed at 864 while the output length is 64. This results in an additional window size $w = 400$. For UK-DALE, the input and output lengths are 464 and 64, respectively. Early stopping is used as a training criteria with a patience of 30. This means that the training procedure stops if the validation performance does not improve for 30 epochs. The detailed training algorithm with our sample augmentation and early stopping is shown in Algorithm 3. We used the MSE loss for the regression subnetwork and the BCE loss for the classification subnetwork. Based on PyTorch 1.6.0, all models are trained via the Adam optimizer with an initial learning rate of 0.001.

Algorithm 3: Training Algorithm with Sample Augmentation (SA) and Early Stopping

```
Input  : Training set $\mathcal{D}$, validation set $\mathcal{D}_{\text{val}}$, number of epochs $E$,
         patience parameter $p$, model $f_\theta$, SA algorithm $\text{Algo}_{\text{SA}}$, SA flag
         $f_{\text{SA}}$
Output: Trained model $f_\theta$
Initialize model parameters;
Set early stopping counter $c = 0$;
Assign a large enough number to the best loss $L_{\text{best}}$;
for $e \in 1, 2, ..., E$ do
    for each batch $\bar{\mathcal{D}} \in \mathcal{D}$ do
        if $f_{\text{SA}} = \mathtt{true}$ then
            $\bar{\mathcal{D}} \leftarrow \text{Algo}_{\text{SA}}(\bar{\mathcal{D}})$; // Algorithm 2
        end
        $\hat{y} \leftarrow f_\theta(\bar{\mathcal{D}})$;
        Compute training loss $L_{\text{train}}$;
        Compute the gradient of loss w.r.t. $\theta$;
        Update parameters in $f_\theta$;
    end
    Compute validation loss $L_{\text{val}}$ on $\mathcal{D}_{\text{val}}$;
    if $L_{\text{val}} < L_{\text{best}}$ then
        Save model $f_\theta$;
        Update $L_{\text{best}} = L_{\text{val}}$;
        Reset $c = 0$;
    end
    else
        $c = c + 1$;
        if $c \geq p$ then
            Stop training;
        end
    end
end
Return the saved model $f_\theta$.
```

5.3.4 Performance Metrics

Mean absolute error (MAE) and signal aggregate error (SAE) are used as evaluation metrics of disaggregation performance for each appliance. They are defined as follows [52]:

$$\text{MAE} = \frac{1}{H} \sum_{t=1}^{H} |y_t - \hat{y}_t| \tag{5.16a}$$

$$\text{SAE} = \frac{1}{S} \sum_{\tau=0}^{S-1} \frac{1}{M} |y_\tau - \hat{y}_\tau| \,, \tag{5.16b}$$

where y_t and $\hat{y}_t$ are the ground truth and estimated power consumption at time t, respectively. H is the length of the test horizon. For SAE, the test horizon is split into S disjoint sub-horizons. The length of each sub-horizon is M. $y_\tau := \sum_{t=1}^{M} y_{M\tau+t}$ is the total power consumption across M timestamps in the τ-th sub-horizon and $\hat{y}_\tau$ is the corresponding estimated value. In this study, for both datasets we set $M = 1200$ and $S = \lfloor \frac{H}{M} \rfloor$ (i.e., the integer part of $\frac{H}{M}$).

The classification performance is evaluated by the F_1 score, which is defined as the harmonic mean of precision and recall:

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{5.17}$$

Precision is the ratio of true positives to the total number of positive predictions. Recall is the ratio of true positives to the total number of actual positive instances. Due to the inherent tradeoff between these two metrics, the F1 score turns out to be especially valuable when dealing with imbalanced datasets.

5.4 Experiment Results

In this section, we showcased the outcomes of three case studies, where the results have been averaged over three independent trials. These case studies demonstrate the effectiveness and merits of our proposed framework. Case 1 evaluates the performance of existing models on a limited labeled data scenario and compares them with our proposed solution that utilizes SA and MATNilm. Case 2 is an ablation study of the proposed approach. Moreover, we compared our SA approach with two state-of-the-art data augmentation methods on four different models: SGN, LDwA, VMA, and MATNilm. Case 3 uses the SGN model to show training performance on a full dataset versus a limited dataset with SA.

5.4.1 Case 1: Performance Comparisons with Limited Training Data

In this test case, we focused on S2 and S3 where only limited training data are available. S2 directly applies a learning model to a limited labeled data scenario while S3 is our proposed solution: MATNilm with SA. Regarding the learning models, both SGN and MATNilm can be implemented with a convolutional or LSTM shared layer. Table 5.2 and Table 5.3 present the comparison of those four model variants. The performance metrics include MAE, SAE and F1 scores. The average improvement for MAE is defined as

$$\text{Imp} = \frac{\text{MAE(SGN)} - \text{MAE(MAT)}}{\text{MAE(SGN)}} \times 100\%\,. \tag{5.18}$$

Similarly, we calculated the average improvements for SAE and F1 scores.

First, our proposed solution MAT-Conv and MAT-LSTM consistently outperform the SGN counterparts in terms of MAE (SAE) scores. For REDD, MAT-Conv and MAT-LSTM models with SA significantly reduce MAE, with an average improvement of 52.23% (58.84%) and 56.04% (63.05%), respectively. We observed similar performance gains for UK-DALE. For the microwave in UK-DALE, the MAE values for different models are at the same level. However, the performance of F1 and SAE is much better for S3.

It is worth noting that there are instances of divergent performance between the MAE and SAE metrics. For example, MAT-Conv has a higher MAE than MAT-LSTM for fridge in REDD, while the SAE performance is the opposite. In such cases, the model having lower SAE is better at accurately disaggregating the appliance's power consumption over time, but is not necessarily very effective in estimating the power at each timestamp.

Second, S3 outperforms S2 in F1 score for both datasets. Recall that S3 has more training samples augmented with appliance activation profiles while S2 lacks representative examples of appliance's power consumption patterns. The limited data scenario makes it harder for the learning models to differentiate between different appliance statuses. This in turn validates the strong capability of our solution to correctly detect on/off status that leads to more accurate disaggregation results. In a nutshell, the results demonstrate that our proposed approach of using MATNilm with SA can substantially improve disaggregation performance, particularly when training data are

Table 5.2: Performance comparisons of existing models and the proposed framework with limited data for REDD dataset. DW, FG, MW, KT, WD are the acronyms for "Dishwasher", "Fridge", "Microwave", "Kettle", "Washer Dryer". Ave and Imp stand for "Average score" and "Average improvement".

Metric	Model	Scenario	DW	FD	MW	WD	Ave	Imp
MAE	SGN - Conv [52]	S2	22.14	39.01	19.40	40.13	30.17	-
	SGN - LSTM	S2	21.98	41.46	21.29	43.91	32.16	-
	MAT - Conv	S3	**8.44**	19.38	13.40	**16.44**	14.41	52.23%
	MAT - LSTM	S3	9.12	**17.86**	**12.49**	17.08	**14.14**	56.04%
SAE	SGN - Conv [52]	S2	21.83	25.78	17.87	39.11	26.15	-
	SGN - LSTM	S2	21.61	31.71	17.89	39.49	27.67	-
	MAT - Conv	S3	**6.94**	**12.30**	10.47	**13.35**	10.76	58.84%
	MAT - LSTM	S3	8.20	12.67	**9.81**	14.29	**10.23**	63.05%
F1	SGN - Conv [52]	S2	0.19	0.80	0.10	0.33	0.36	-
	SGN - LSTM	S2	0.23	0.71	0.08	0.61	0.41	-
	MAT - Conv	S3	0.80	0.88	0.66	**0.67**	0.75	110.98%
	MAT - LSTM	S3	**0.82**	**0.91**	**0.74**	0.64	**0.78**	89.23%

limited.

Finally, Fig. 5.2 shows the convergence of the MAT-conv model. It can be seen that the total loss and all individual appliance-level losses converge in about 80 epochs.

5.4.2 Case 2: Ablation Study and DA comparisons

Table 5.4 summarizes the MAE scores obtained by the ablation study for the sample augmentation and MATNilm with convolutional layers as the encoder. The study evaluates the performance of the framework with different combinations of SGN, sample augmentation (SA), multi-appliance task (MT), temporal attention (TA), and appliance attention (AA). The results show that SA significantly reduces the MAE values. The errors further decrease by incorporating the temporal and appliance attention

Table 5.3: Performance comparisons of existing models and the proposed framework with limited data for UK-DALE dataset. DW, FG, MW, KT, WD are the acronyms for “Dishwasher”, “Fridge”, “Microwave”, “Kettle”, “Washer Dryer”. Ave and Imp stand for “Average score” and “Average improvement”.

Metric	Model	Scenario	DW	FD	MW	KT	WD	Ave	Imp
MAE	SGN - Conv [52]	S2	19.81	27.05	8.03	15.09	20.92	18.18	-
	SGN - LSTM	S2	21.40	32.35	7.90	7.22	21.24	18.02	-
	MAT - Conv	S3	10.88	17.06	**7.08**	5.95	6.52	9.50	47.75%
	MAT - LSTM	S3	**6.51**	**15.86**	8.20	**5.36**	**5.37**	**8.26**	54.17%
SAE	SGN - Conv [52]	S2	15.00	13.07	8.06	13.09	20.79	14.00	-
	SGN - LSTM	S2	19.52	16.23	7.49	5.49	20.24	13.79	-
	MAT - Conv	S3	9.44	6.93	**5.53**	**3.01**	4.54	5.89	57.92%
	MAT - LSTM	S3	**5.39**	**6.79**	6.70	3.34	**3.65**	**5.18**	62.48%
F1	SGN - Conv [52]	S2	0.85	0.65	0.00	0.39	0.08	0.39	-
	SGN - LSTM	S2	0.76	0.60	0.18	0.85	0.09	0.50	-
	MAT - Conv	S3	0.82	0.83	**0.67**	**0.91**	0.79	0.80	104.41%
	MAT - LSTM	S3	**0.91**	**0.84**	0.65	0.90	**0.82**	**0.83**	66.76%

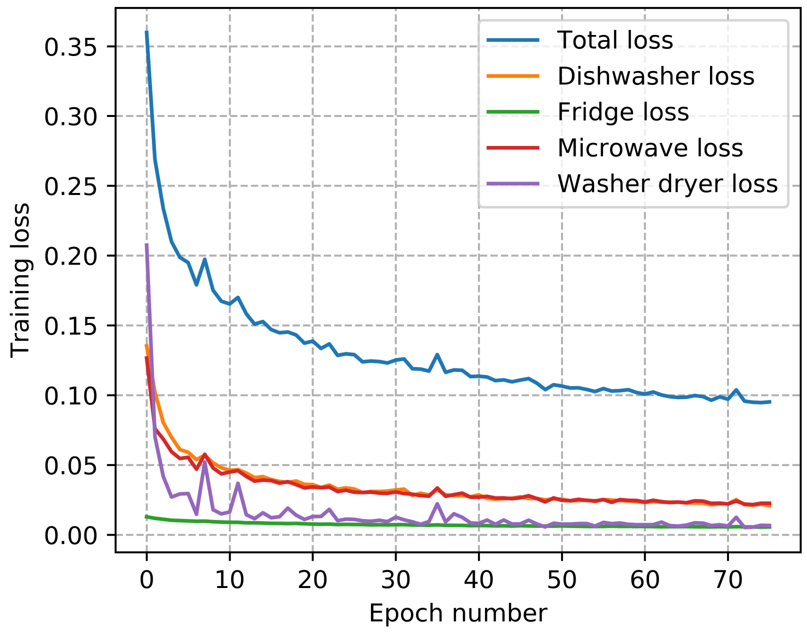


Figure 5.2: The convergence of MAT-conv training losses.

to the vanilla MT network. This validates the effectiveness of each proposed module in our framework.

Using four models, namely SGN [52], LDwA [53], VMA [65], and MATNilm, we conducted further comparisons between our sample augmentation approach and two state-of-the-art NILM data augmentation methods [68] and [69]. However, since both of these competing alternatives were not originally designed to address the limited data scenario or the multi-appliance approach, certain modifications were necessary to tailor them to our specific problem setting. The original data augmentation methods were designed to augment a single target appliance and generated (aggregate, appliance) pairs of signals for each appliance. This resulted in misalignment among the aggregate signals and rendered them unsuitable for the multi-appliance approach, where the aggregate signal remains the same for each appliance. To ensure a fair comparison, we employed an aligned aggregate signal dataset for both the single-appliance and multi-appliance models. To adapt Kong's method [68], we duplicated the one-day labeled data 20 times and applied the same augmentation steps to all target appliances, following the reference algorithms in [68]. For [69], we first duplicated the one-day labeled data three times. Then, we generated an augmented synthetic dataset for each appliance and concatenated them to create the final dataset for training and evaluation. These modifications allowed us to compare the performance of our sample augmentation approach against these state-of-the-art methods in a fair and consistent manner.

Table 5.5 summarizes the MAE performance for the aforementioned comparisons. Notably, our proposed MATNilm with SA achieves the lowest MAE scores for all appliances, with an average MAE of 14.41. When considering a single model, our proposed SA demonstrates significant performance improvement over the other two competing data augmentation (DA) methods [68, 69], except for VMA. This suggests that for deep models, SA is capable of generating richer data that benefits the model's learning process. However, in the case of VMA, which has a shallow model structure, its learning ability is limited, and therefore it does not derive significant benefits from the augmented training data.

Without the two-dimensional attention, MAT and VMA have inferior performance compared with the single-appliance SGN. A possible reason is that the shared and split structures of the two multi-appliance approaches cannot adequately capture the complex cross-appliance information, which is necessary for yielding better disaggregation outcomes. Furthermore, employing a single model for all appliances does not offer much flexibility when it comes to fine-tuning the best model for each appliance.

To sum up, the numerical results indicate that SA and MAT with 2DMA are crucial components for achieving accurate appliance-level energy disaggregation. The ablation study provides us an insight into the relative importance of different modules of the proposed framework.

5.4.3 Case 3: Comparison with Training on Full Dataset

In practice, obtaining household-labeled data is often challenging. Appliance operation profiles are comparatively easier to get. Hence, we aimed to investigate whether the model can achieve comparable performance with limited labeled data and

Table 5.4: REDD: MAE results for ablation study of the proposed framework. MT, TA, AA are the acronyms for "Multi-appliance-task", "Temporal attention", "Appliance attention".

SGN	SA	MT	TA	AA	DW	FG	MW	WD
✓					22.14	39.01	19.40	40.13
✓	✓				14.31	28.68	16.90	21.79
✓	✓	✓			24.26	42.83	19.78	34.18
✓	✓	✓	✓		9.73	20.08	**13.33**	20.12
✓	✓	✓	✓	✓	**8.44**	**19.38**	13.40	**16.44**

Table 5.5: REDD: MAE results for performance comparison between the proposed SA and two existing augmentation methods.

DA	Model	DW	FD	MW	WD	Average
Kong[68]	SGN[52]	22.14	39.01	19.40	40.13	30.17
	LDwA[53]	17.62	39.44	18.93	39.21	28.80
	VMA[65]	26.53	42.14	29.85	49.96	37.12
	MATNilm	11.72	32.61	26.93	46.66	29.48
Rafiq[69]	SGN[52]	21.30	42.81	19.77	33.29	29.29
	LDwA[53]	22.14	39.01	19.40	40.13	30.17
	VMA[65]	24.59	49.62	18.66	41.59	33.61
	MATNilm	24.77	44.68	20.12	42.31	32.97
SA	SGN[52]	14.31	28.68	16.90	21.79	20.42
	LDwA[53]	16.41	28.32	19.34	27.63	22.93
	VMA[65]	25.41	47.69	18.47	45.37	34.24
	MATNilm	**8.44**	**19.38**	**13.40**	**16.44**	**14.41**

appliance operation profiles, compared with the case of several weeks of data from multiple houses. Specifically, we compared the results of models trained in S1 and S3. The latter uses only one day of labeled data and appliance operation profiles, which are extracted from the same training houses as in S1, for sample augmentation.

Table 5.6 presents the performance comparison in scenarios S1 and S3 for the REDD dataset. The results indicate that limited data with SA can significantly im-

Table 5.6: REDD: Comparison of the SGN model trained on the full dataset (S1) versus on the limited dataset with SA (S3).

Metric	Scenario	DW	FD	MW	WD	Average
MAE	S1	19.51	**27.27**	23.61	39.22	27.40
	S3	**14.31**	28.68	**16.90**	**21.79**	**20.42**
SAE	S1	18.92	17.23	18.33	34.79	22.32
	S3	**12.73**	**16.77**	**11.94**	**16.12**	**14.39**
F1	S1	0.34	0.83	0.04	0.29	0.38
	S3	**0.70**	**0.85**	**0.65**	**0.59**	**0.70**

prove the F1 score for most appliances, except for the fridge. This improvement can be attributed to the fact that SA tends to balance the on/off status for each appliance, which also results in better performance in terms of MAE and SAE. However, since the fridge operates quite uniformly over time, there is no significant improvement in its performance with SA.

5.5 Summary

In this chapter, a solution for the NILM problem with limited training data is designed. Specifically, the multi-appliance-task framework adopts a shared-hierarchical split structure for each appliance based on its power consumption (regression) and status (classification) tasks. A two-dimensional attention mechanism is developed to capture power consumption relationships across each appliance and time step. Instead of generating a time-continuously synthetic dataset, we designed a dynamic sample augmentation algorithm, where augmented training samples are randomly generated for each appliance in each mini-batch. Experiment results show a significant performance boost by the proposed solution. Moreover, the simulation results also reveal the importance of collecting individual appliance operation profiles, which may open up a new research direction for the NILM.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we first developed an attention-based neural load forecasting model for multi-horizon STLF. The proposed approach includes a dynamic feature selection layer combined with a BiLSTM encoder to extract relevant features. A BiLSTM decoder with a hierarchical temporal attention layer is then used to decode the next day's load based on future input features. The temporal attention layer provides a systematic way to incorporate similar day information. Extensive simulation results demonstrate the effectiveness of the proposed approach, which outperforms state-of-the-art benchmarks.

Next, a unified deep learning framework is proposed for load forecasting, which incorporates time-varying feature weighting, a load forecasting model with hierarchical temporal attention, and feature-reinforced error correction. This framework is designed with modularity in mind, allowing for good generalization capabilities. Firstly, the feature-weighting mechanism assigns temporal weights to input features. Secondly, a recurrent encoder-decoder structure with hierarchical attention is developed to serve as a load predictor. The hierarchical attention facilitates the selection of similar days, allowing for a re-evaluation of the importance of historical information at each time step. Lastly, an error correction module is introduced to leverage the errors and learned hidden information of features, further enhancing the forecasting performance of the model. This framework offers an effective solution to the electric load forecasting problem and can be easily adapted to various other forecasting tasks.

Finally, a solution has been developed to address the NILM problem when there is limited training data available. The solution utilizes a multi-appliance-task framework that employs a shared-hierarchical split structure for each appliance, taking into account both power consumption (regression) and status (classification) tasks. To capture the relationships between power consumption and time steps, a two-dimensional attention mechanism has been devised. Instead of generating a synthetic dataset continuously over time, a dynamic sample augmentation algorithm has been designed. This algorithm randomly generates augmented training samples for each appliance in each mini-batch. Experimental results demonstrate a significant performance improvement achieved by the proposed solution. Additionally, simulation results highlight the importance of collecting individual appliance operation profiles, which could potentially open up new avenues for research in the field of NILM.

6.1.1 Contributions

This thesis offers the following notable contributions:

1. We introduced a novel attention-based dynamic feature selection mechanism for STLF, emphasizing the capability of weighing each feature concerning the entire input sequence and integrating a hierarchical temporal attention layer for similar day and hour information.

2. We proposed a unifying framework for multi-horizon STLF. The feature weighting is extracted from the encoder, enhancing the modularity of the design and improving interpretability. The proposed error correction module utilizes transfer learning, eliminating the need for a new model design while inheriting the learned feature knowledge.

3. We developed a solution for the NILM problem under the constraints of limited training data. Through a dynamic sample augmentation algorithm, we demonstrated competitive performance against models trained on comprehensive datasets. A multi-appliance-task model with a two-dimensional multi-head self-attention mechanism further enhanced the efficacy of our solution by capturing power consumption relationships across multiple appliances and time steps.

6.2 Future work

Current efforts in load forecasting are primarily focused on point prediction, which provides a specific expected value for future demand. However, this approach does not consider various uncertainties such as changes in weather, unexpected events, or consumer behavior. Probabilistic load forecasting, in contrast, offers a range of possible outcomes, giving a more comprehensive view of potential future uncertainties. Prospective paths in load forecasting may utilize methods like conformal prediction to generate interval predictions. Additionally, incorporating real-time adjustments into load forecasts could be a pivotal step. Such dynamic corrections would enable the forecasting model to adapt to immediate changes or new information, maintaining its relevance and accuracy. Moreover, regularly updating forecasting models with the latest datasets will guarantee that the predictions remain up-to-date and accurately capture the evolving trends in electricity consumption. Finally, it is beneficial to incorporate feature errors by using total least-squares methods [89].

Deep learning models heavily depend on large and diverse training datasets to achieve optimal generalization capabilities. The challenge is amplified in the context of non-intrusive load monitoring (NILM). The total energy consumption is a combination of the outputs from different appliances. By collecting the operation profile of each appliance instead of just the labeled data of the aggregate appliance pairs, we can obtain a more comprehensive and arguably better dataset for training deep learning models. The potential of appliance-level data goes beyond individual patterns. By utilizing a wide range of such data, it becomes possible to replicate numerous household scenarios. This replication can effectively simulate real-life conditions, allowing the model to train in different circumstances, ultimately improving its resilience and flexibility.

When creating training samples, it is important to determine specific hyperparameters. As explained in Chapter 5, variables such as the sample augmentation probability for each appliance (represented as prob^i) and the modify-mode probability mass function for each appliance (represented as p^i) have been pre-defined manually. While this method is functional, it can introduce biases and may not be optimally efficient.

Automated sample augmentation is a promising area for future investigations in NILM. Rather than relying on manual presets, an algorithm could iteratively refine hyperparameters and autonomously optimize for the best results. This Auto Sample Augmentation mechanism could reduce manual interventions and enhance the adaptability and precision of the model across various household scenarios. Furthermore, integrating newer architectures and methodologies into deep learning paradigms could further refine NILM outcomes. Meta-learning/few-shot learning, for instance, can be explored to accelerate the training process and achieve better generalization with limited data. With these methodologies, we expect the NILM system can quickly adapt to new or rarely seen appliances or houses via limited training data, which will be practical in real-world situations where new devices are introduced.

6.2.1 Literature Review

In the field of deep learning, Data Augmentation (DA) has become an essential technique for improving model performance, especially when dealing with limited training datasets. However, traditional DA methods often require manual design and extensive domain knowledge, making them time-consuming and not always optimal. To overcome these limitations, there has been a growing demand for Automated Data Augmentation (AutoDA) techniques.

AutoDA is a data-driven approach that is capable of identifying superior data augmentation strategies. AutoDA techniques can be broadly classified into three categories based on their augmentation synthesis mechanisms: composition-based, mixing-based, and generation-based AutoDA [90].

Composition-based methods aim to improve model generalization by integrating various data transformation operations. One notable example is AutoAugment [91], which uses a reinforcement learning framework to learn optimal data augmentation operations like shearing and rotation to maximize validation accuracy. However, AutoAugment can be computationally expensive. RandAugment addresses this issue by fine-tuning the search space of AutoAugment and calibrating multiple transformations via a singular magnitude parameter [92]. Population-Based Augmentation (PBA) combines AutoAugment with population-based training (PBT) principles, using an evolutionary algorithm to train multiple models simultaneously and improve augmentation techniques iteratively [93].

Mixing-based techniques in machine learning involve creating new training samples by blending existing data. Two popular techniques in this category are Mixup and Cutmix. Mixup generates a weighted fusion of two pre-existing images [94], while Cutmix creates a unique image by combining a segment from one image with fragments from another [95]. Another technique in this category is MoCHi, which leverages a catalog of negative features to create synthetic, challenging instances in a structured manner [96].

The generation-based approach focuses on creating new data entities, utilizing Generative AI algorithms as data catalysts. Computer vision is enhanced through the use of tools such as GANs, VAEs, and diffusion models, which are skilled at generating new training samples [97]. Additionally, the neural style transfer technique is highly beneficial in diversifying datasets by producing images that combine different styles while preserving the essence of the content [98]. Meanwhile, Large Language Models (LLMs) have revolutionized natural language processing, enabling the creation of new textual samples and serving as advanced annotators [99].

6.2.2 Problem formulation

A sample augmentation policy includes N_a appliance selection policies: $\{\bar{s}^i(\bar{O}^i; prob^i) \mid 1 \leq i \leq N_a\}$ which are applied in sequence.

$$\bar{s}^i(\bar{O}^i; prob^i) = \begin{cases} s^i\left(\bar{O}^i\right) & : \text{ with probability } prob^i, \\ \mathbf{0} & : \text{ with probability } 1 - prob^i. \end{cases}$$

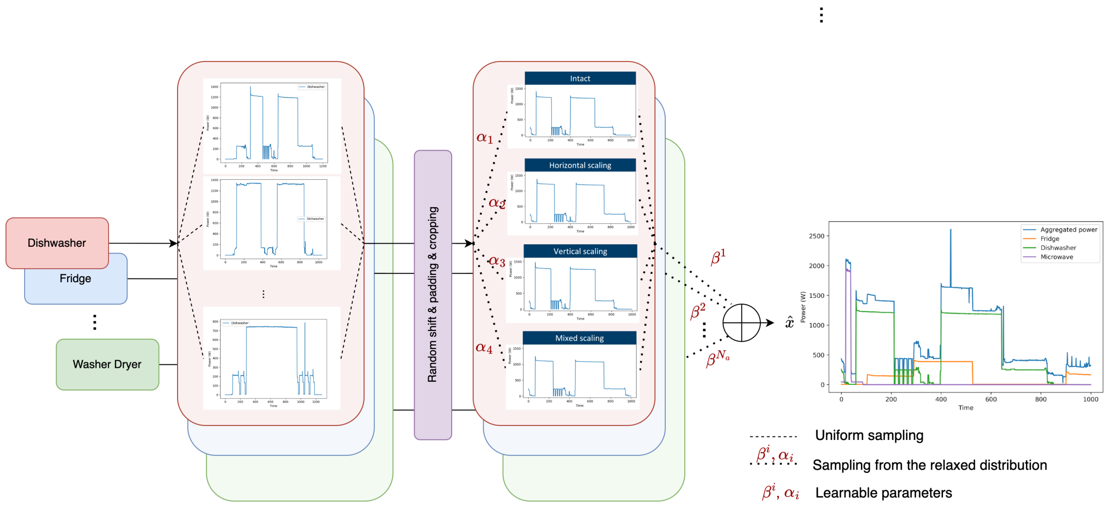


Figure 6.1: Differentiable policy structure for AutoNILM.

Each appliance selection policy s includes K operations (e.g. intact, horizontal scaling, vertical scaling and mixing scaling) $\bar{O}_k^i\left(\tilde{\mathbf{y}}_j^i; w_k^i, m_k^i\right)$ $(1 \le k \le K)$ which will be sampled from a categorical distribution. $\tilde{\mathbf{y}}_j^i$ is uniformly sampled from $\mathcal{A}_i$. Operation $\bar{O}_k^i$ is applied to the original appliance profile $\tilde{\mathbf{y}}_j^i$ with two continuous parameters: w_k^i (the probability of applying the operation) and m_k^i (the magnitude of the operation):

Therefore, augmented aggregate input is given by:

$$\hat{\mathbf{x}} = \sum_{i=0}^{i=N_a} s^i(\bar{O}^i; prob^i) \tag{6.1}$$

This approach involves modeling appliance selection and operation using Bernoulli and Categorical distributions, respectively. Then, the Monte Carlo gradient estimate problem can be utilized to optimize sample augmentation from two distributions. Future work includes optimizing the sample augmentation parameters by using differentiable relaxation of the Bernoulli and Categorical distribution and gradient estimator. The procedure is shown in Fig. 6.2.

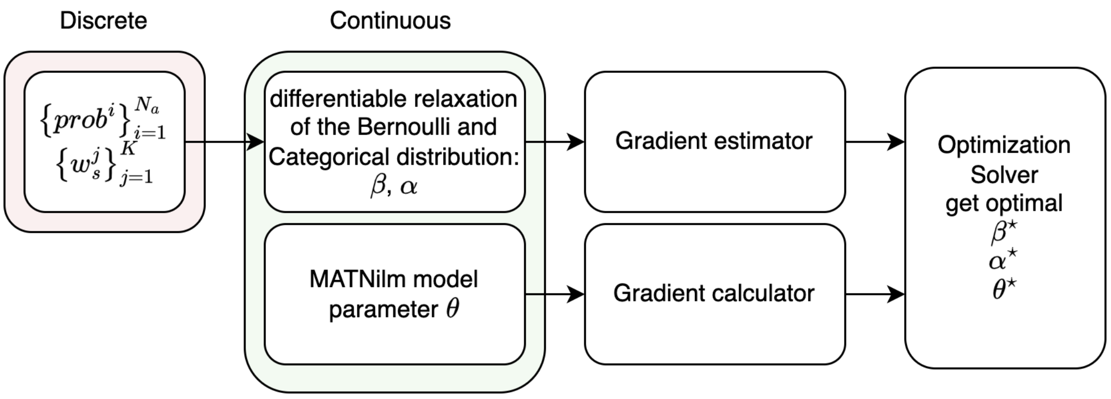

Discrete
Continuous
$\{prob^i\}_{i=1}^{N_a}$
$\{w_s^j\}_{j=1}^{K}$
differentiable relaxation of the Bernoulli and Categorical distribution: β, α
MATNilm model parameter θ
Gradient estimator
Gradient calculator
Optimization Solver get optimal $\beta^\star$ $\alpha^\star$ $\theta^\star$

Figure 6.2: Workflow of AutoNILM.

References

[1] U. DOE, "Grid 2030: A national vision for electricity's second 100 years," *US DOE Report*, pp. 137–140, 2003.

[2] Y. Peng, Y. Wang, X. Lu, H. Li, D. Shi, Z. Wang, and J. Li, "Short-term load forecasting at different aggregation levels with predictability analysis," in *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, pp. 3385–3390, IEEE, 2019.

[3] C. Feng, M. Sun, and J. Zhang, "Reinforced deterministic and probabilistic load forecasting via Q-learning dynamic model selection," *IEEE Transactions on Smart Grid*, vol. 11, pp. 1377–1386, Mar. 2020.

[4] X. Kong, C. Li, C. Wang, Y. Zhang, and J. Zhang, "Short-term electrical load forecasting based on error correction using dynamic mode decomposition," *Applied Energy*, vol. 261, p. 114368, 2020.

[5] S. Makonin, Z. J. Wang, and C. Tumpach, "Rae: The rainforest automation energy dataset for smart grid meter data analysis," *data*, vol. 3, no. 1, p. 8, 2018.

[6] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, pp. 55–75, Jul. 2018.

[7] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning

for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, Feb. 2018.

[8] G. Mbamalu and M. El-Hawary, "Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation," *IEEE Transactions on Power Systems*, vol. 8, no. 1, pp. 343–348, 1993.

[9] J.-F. Chen, W.-M. Wang, and C.-M. Huang, "Analysis of an adaptive time-series autoregressive moving-average (arma) model for short-term load forecasting," *Electric Power Systems Research*, vol. 34, no. 3, pp. 187–196, 1995.

[10] G. Juberias, R. Yunta, J. G. Moreno, and C. Mendivil, "A new arima model for hourly load forecasting," in *1999 IEEE Transmission and Distribution Conference (Cat. No. 99CH36333)*, vol. 1, pp. 314–319, IEEE, 1999.

[11] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, pp. 1014–1020, Jul. 2003.

[12] E. Ceperic, V. Ceperic, and A. Baric, "A strategy for short-term load forecasting by support vector regression machines," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4356–4364, 2013.

[13] G. Dudek, "Short-term load forecasting using random forests," in *Intelligent Systems' 2014*, pp. 821–828, Springer, 2015.

[14] Y.-Y. Cheng, P. P. Chan, and Z.-W. Qiu, “Random forest based ensemble system for short term load forecasting,” in *2012 international conference on machine learning and cybernetics*, vol. 1, pp. 52–56, IEEE, 2012.

[15] S. B. Taieb and R. J. Hyndman, “A gradient boosting approach to the kaggle load forecasting competition,” *International journal of forecasting*, vol. 30, no. 2, pp. 382–394, 2014.

[16] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, “A dual-stage attention-based recurrent neural network for time series prediction,” *Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 2627–2633, Aug. 2017.

[17] Q. Cao, B. T. Ewing, and M. A. Thompson, “Forecasting wind speed with recurrent neural networks,” *European Journal of Operational Research*, vol. 221, pp. 148–154, Aug. 2012.

[18] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, Mar. 1994.

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, Dec. 2017.

[23] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, Oct. 2014.

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

[25] D. Panday, R. C. de Amorim, and P. Lane, "Feature weighting as a tool for unsupervised feature selection," *Information processing letters*, vol. 129, pp. 44–52, 2018.

[26] Y. Xuan, W. Si, J. Zhu, Z. Sun, J. Zhao, M. Xu, and S. Xu, "Multi-model fusion short-term load forecasting based on random forest feature selection and hybrid neural network," *IEEE Access*, vol. 9, pp. 69002–69009, 2021.

[27] Y. Deng, B. Wang, and Z. Lu, "A hybrid model based on data preprocessing strat-

egy and error correction system for wind speed forecasting," *Energy Conversion and Management*, vol. 212, p. 112779, 2020.

[28] J. Duan, H. Zuo, Y. Bai, J. Duan, M. Chang, and B. Chen, "Short-term wind speed forecasting using recurrent neural networks with error correction," *Energy*, vol. 217, p. 119397, 2021.

[29] H. Liu and C. Chen, "Multi-objective data-ensemble wind speed forecasting model with stacked sparse autoencoder and adaptive decomposition-based error correction," *Applied Energy*, vol. 254, p. 113686, 2019.

[30] Z. Li, L. Ye, Y. Zhao, X. Song, J. Teng, and J. Jin, "Short-term wind power prediction based on extreme learning machine with error correction," *Protection and Control of Modern Power Systems*, vol. 1, no. 1, pp. 1–8, 2016.

[31] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[32] L. Cai, J. Gu, and Z. Jin, "Two-layer transfer-learning-based architecture for short-term load forecasting," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1722–1732, 2019.

[33] D. Wu, B. Wang, D. Precup, and B. Boulet, "Multiple kernel learning-based transfer regression for electric load forecasting," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1183–1192, 2019.

[34] E. Lee and W. Rhee, "Individualized short-term electric load forecasting with deep

neural network based transfer learning and meta learning," *IEEE Access*, vol. 9, pp. 15413–15425, 2021.

[35] S. S. Hosseini, K. Agbossou, S. Kelouwani, and A. Cardenas, "Non-intrusive load monitoring through home energy management systems: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 79, pp. 1266–1274, 2017.

[36] Y.-H. Lin and M.-S. Tsai, "Development of an improved time–frequency analysis-based nonintrusive load monitor for load demand identification," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 6, pp. 1470–1483, 2013.

[37] Y. Liu, X. Wang, and W. You, "Non-intrusive load monitoring by voltage–current trajectory enabled transfer learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5609–5619, 2018.

[38] A. L. Wang, B. X. Chen, C. G. Wang, and D. Hua, "Non-intrusive load monitoring algorithm based on features of v–i trajectory," *Electric Power Systems Research*, vol. 157, pp. 134–144, 2018.

[39] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.

[40] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models," *Machine learning*, vol. 29, no. 2, pp. 245–273, 1997.

[41] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggre-

gation of low frequency power measurements," in *Proceedings of the 2011 SIAM international conference on data mining*, pp. 747–758, SIAM, 2011.

[42] R. Bonfigli, S. Squartini, M. Fagiani, and F. Piazza, "Unsupervised algorithms for non-intrusive load monitoring: An up-to-date overview," in *2015 IEEE 15th international conference on environment and electrical engineering (EEEIC)*, pp. 1175–1180, IEEE, 2015.

[43] M. Zhuang, M. Shahidehpour, and Z. Li, "An overview of non-intrusive load monitoring: Approaches, business applications, and challenges," in *2018 International Conference on Power System Technology (POWERCON)*, pp. 4291–4299, IEEE, 2018.

[44] B. Najafi, S. Moaveninejad, and F. Rinaldi, "Data analytics for energy disaggregation: Methods and applications," in *Big data application in power systems*, pp. 377–408, Elsevier, 2018.

[45] H. Bousbiat, A. Faustine, C. Klemenjak, L. Pereira, and W. Elmenreich, "Unlocking the full potential of neural nilm: On automation, hyperparameters & modular pipelines," *IEEE Transactions on Industrial Informatics*, 2022.

[46] A. Moradzadeh, B. Mohammadi-Ivatloo, M. Abapour, A. Anvari-Moghaddam, S. Gholami Farkoush, and S.-B. Rhee, "A practical solution based on convolutional neural network for non-intrusive load monitoring," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9775–9789, 2021.

[47] C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton, "Sequence-to-point learning with neural networks for non-intrusive load monitoring," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[48] Y. Liu, J. Qiu, and J. Ma, "Samnet: Toward latency-free non-intrusive load monitoring via multi-task deep learning," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2412–2424, 2021.

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

[50] J. Feng, K. Li, H. Zhang, X. Zhang, and Y. Yao, "Multi-channel spatio-temporal feature fusion method for nilm," *IEEE Transactions on Industrial Informatics*, 2022.

[51] G. Tanoni, E. Principi, and S. Squartini, "Multilabel appliance classification with weakly labeled data for non-intrusive load monitoring," *IEEE Transactions on Smart Grid*, vol. 14, no. 1, pp. 440–452, 2022.

[52] C. Shin, S. Joo, J. Yim, H. Lee, T. Moon, and W. Rhee, "Subtask gated networks for non-intrusive load monitoring," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1150–1157, 2019.

[53] V. Piccialli and A. M. Sudoso, "Improving non-intrusive load disaggregation

through an attention-based deep neural network," *Energies*, vol. 14, no. 4, p. 847, 2021.

[54] K. Chen, Y. Zhang, Q. Wang, J. Hu, H. Fan, and J. He, "Scale- and context-aware convolutional non-intrusive load monitoring," *IEEE Transactions on Power Systems*, vol. 35, pp. 2362–2373, May 2020.

[55] M. Kaselimi, N. Doulamis, A. Voulodimos, A. Doulamis, and E. Protopapadakis, "Energan++: A generative adversarial gated recurrent network for robust energy disaggregation," *IEEE Open Journal of Signal Processing*, vol. 2, pp. 1–16, 2020.

[56] Y. Liu, L. Zhong, J. Qiu, J. Lu, and W. Wang, "Unsupervised domain adaptation for nonintrusive load monitoring via adversarial and joint adaptation network," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 266–277, 2021.

[57] A. Harell, S. Makonin, and I. V. Bajić, "Wavenilm: A causal neural network for power disaggregation from the complex power signal," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8335–8339, IEEE, 2019.

[58] B. Liu, W. Luan, and Y. Yu, "Dynamic time warping based non-intrusive load transient identification," *Applied energy*, vol. 195, pp. 634–645, 2017.

[59] A. Rahimpour, H. Qi, D. Fugate, and T. Kuruganti, "Non-intrusive energy disaggregation using non-negative matrix factorization with sum-to-k constraint," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4430–4441, 2017.

[60] P. A. Schirmer and I. Mporas, “Non-intrusive load monitoring: A review,” *IEEE Transactions on Smart Grid*, 2022.

[61] Y. Yang, J. Zhong, W. Li, T. A. Gulliver, and S. Li, “Semisupervised multilabel deep learning based nonintrusive load monitoring in smart grids,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 6892–6902, 2019.

[62] J. A. Mueller and J. W. Kimball, “Accurate energy use estimation for nonintrusive load monitoring in systems of known devices,” *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2797–2808, 2016.

[63] Y. Liu, G. Geng, S. Gao, and W. Xu, “Non-intrusive energy use monitoring for a group of electrical appliances,” *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3801–3810, 2016.

[64] J. Kelly and W. Knottenbelt, “Neural nilm: Deep neural networks applied to energy disaggregation,” in *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, pp. 55–64, 2015.

[65] P. A. Schirmer and I. Mporas, “Binary versus multiclass deep learning modelling in energy disaggregation,” in *Energy and Sustainable Futures: Proceedings of 2nd ICESF 2020*, pp. 45–51, Springer International Publishing Cham, 2021.

[66] A. Faustine, L. Pereira, H. Bousbiat, and S. Kulkarni, “Unet-nilm: A deep neural network for multi-tasks appliances state detection and power estimation in nilm,” in

Proceedings of the 5th International Workshop on Non-Intrusive Load Monitoring, pp. 84–88, 2020.

[67] D. Li, J. Li, X. Zeng, V. Stankovic, L. Stankovic, C. Xiao, and Q. Shi, "Transfer learning for multi-objective non-intrusive load monitoring in smart building," *Applied Energy*, vol. 329, p. 120223, 2023.

[68] W. Kong, Z. Y. Dong, B. Wang, J. Zhao, and J. Huang, "A practical solution for non-intrusive type ii load monitoring based on deep learning and post-processing," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 148–160, 2019.

[69] H. Rafiq, X. Shi, H. Zhang, H. Li, M. K. Ochani, and A. A. Shah, "Generalizability improvement of deep learning-based non-intrusive load monitoring system using data augmentation," *IEEE Transactions on Smart Grid*, vol. 12, no. 4, pp. 3265–3277, 2021.

[70] H. Cimen, Y. Wu, Y. Wu, Y. Terriche, J. C. Vasquez, and J. M. Guerrero, "Deep learning-based probabilistic autoencoder for residential energy disaggregation: An adversarial approach," *IEEE Transactions on Industrial Informatics*, 2022.

[71] D. Li and S. Dick, "Residential household non-intrusive load monitoring via graph-based multi-label semi-supervised learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 4615–4627, 2018.

[72] Z. Zhou, Y. Xiang, H. Xu, Z. Yi, D. Shi, and Z. Wang, "A novel transfer learning-

based intelligent nonintrusive load-monitoring with limited measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–8, 2020.

[73] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang, "Multi-horizon time series forecasting with temporal attention learning," *International Conference on Knowledge Discovery & Data Mining*, pp. 2527–2535, Jul. 2019.

[74] C. Feng and J. Zhang, "Hourly-similarity based solar forecasting using multi-model machine learning blending," *2018 IEEE Power Energy Society General Meeting*, pp. 1–5, Aug. 2018.

[75] M. Barman, N. D. Choudhury, and S. Sutradhar, "A regional hybrid GOA-SVM model based on similar day approach for short-term load forecasting in assam, india," *Energy*, vol. 145, pp. 710–720, Feb. 2018.

[76] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *International Journal of Forecasting*, vol. 32, pp. 896–913, Jul.–Sept. 2016.

[77] Y. Saeys, T. Abeel, and Y. Van de Peer, "Robust feature selection using ensemble feature selection techniques," *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 313–325, Sept. 2008.

[78] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky,

and V. A. Kamaev, “A survey of forecast error measures,” *World Applied Sciences Journal*, vol. 24, pp. 171–176, Jan. 2013.

[79] N. Kwak and C.-H. Choi, “Input feature selection for classification problems,” *IEEE Transactions on Neural Networks*, vol. 13, pp. 143–159, Aug. 2002.

[80] J. Xiong, P. Zhou, A. Chen, and Y. Zhang, “Attention-based neural load forecasting: A dynamic feature selection approach,” in *2021 IEEE Power & Energy Society General Meeting*, pp. 1–5, IEEE, 2021.

[81] A. Dedinec, S. Filiposka, A. Dedinec, and L. Kocarev, “Deep belief network based electricity load forecasting: An analysis of macedonian case,” *Energy*, vol. 115, pp. 1688–1700, 2016.

[82] M. Khodayar, O. Kaynak, and M. E. Khodayar, “Rough deep neural architecture for short-term wind speed forecasting,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 2770–2779, 2017.

[83] S. H. Rafi, S. R. Deeba, E. Hossain, *et al.*, “A short-term load forecasting method using integrated cnn and lstm network,” *IEEE Access*, vol. 9, pp. 32436–32448, 2021.

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

[85] N. Ghadimi, A. Akbarimajd, H. Shayeghi, and O. Abedinia, “Two stage forecast

engine with feature selection technique and improved meta-heuristic algorithm for electricity load forecasting," *Energy*, vol. 161, pp. 130–142, 2018.

[86] J. Z. Kolter and M. J. Johnson, "REDD: A public data set for energy disaggregation research," in *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA*, vol. 25, pp. 59–62. Issue: Citeseer.

[87] 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, 2015.

[88] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, "Nilmtk: An open source toolkit for non-intrusive load monitoring," in *Proceedings of the 5th international conference on Future energy systems*, pp. 265–276, 2014.

[89] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal processing*, vol. 87, no. 10, pp. 2283–2302, 2007.

[90] T.-H. Cheung and D.-Y. Yeung, "A survey of automated data augmentation for image classification: Learning to compose, mix, and generate," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[91] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123, 2019.

[92] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “Randaugment: Practical automated data augmentation with a reduced search space,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 702–703, 2020.

[93] D. Ho, E. Liang, X. Chen, I. Stoica, and P. Abbeel, “Population based augmentation: Efficient learning of augmentation policy schedules,” in *International conference on machine learning*, pp. 2731–2741, PMLR, 2019.

[94] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017.

[95] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

[96] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, “Hard negative mixing for contrastive learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 21798–21809, 2020.

[97] A. Antoniou, A. Storkey, and H. Edwards, “Data augmentation generative adversarial networks,” *arXiv preprint arXiv:1711.04340*, 2017.

[98] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.

[99] V. Kumar, A. Choudhary, and E. Cho, “Data augmentation using pre-trained transformer models,” *arXiv preprint arXiv:2003.02245*, 2020.