

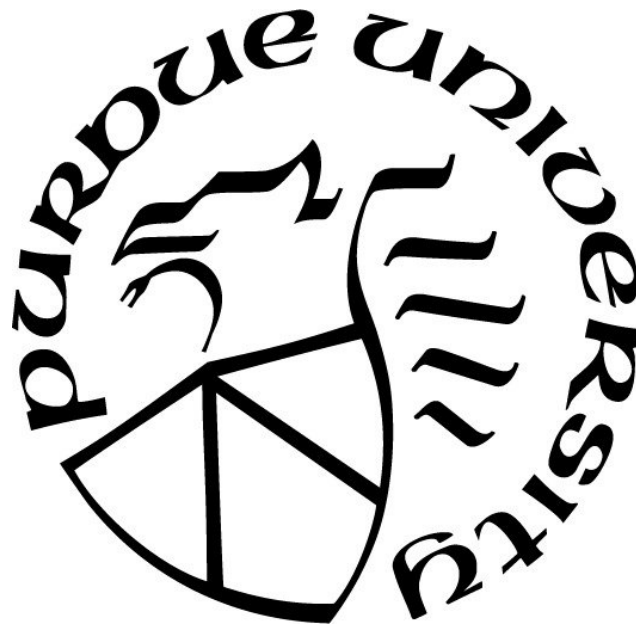
**NON-INTRUSIVE LOAD EXTRACTION OF ELECTRIC VEHICLE  
CHARGING LOADS FOR EDGE COMPUTING**

by  
**Hyeonae Jang**

**A Thesis**

*Submitted to the Faculty of Purdue University  
In Partial Fulfillment of the Requirements for the Degree of*

**Master of Science**



Department of Computer and Information Technology

West Lafayette, Indiana

May 2020

**THE PURDUE UNIVERSITY GRADUATE SCHOOL**  
**STATEMENT OF COMMITTEE APPROVAL**

Dr. Eric T. Matson, Chair

Department of Computer and Information Technology

Dr. John A. Springer

Department of Computer and Information Technology

Dr. Rajesh Sankaran

Mathematics and Computer Science Division, Argonne National Laboratory

**Approved by:**

Dr. Eric T. Matson

Head of the Graduate Program

## TABLE OF CONTENTS

LIST OF TABLES . . . . .	6
LIST OF FIGURES . . . . .	7
LIST OF ABBREVIATIONS . . . . .	9
GLOSSARY . . . . .	10
ABSTRACT . . . . .	11
CHAPTER 1. INTRODUCTION . . . . .	12
1.1 Statement of Problem . . . . .	12
1.2 Significance . . . . .	14
1.3 Research Question . . . . .	14
1.4 Assumptions . . . . .	15
1.5 Limitations . . . . .	15
1.6 Delimitations . . . . .	15
1.7 Summary . . . . .	16
CHAPTER 2. REVIEW OF LITERATURE . . . . .	17
2.1 Edge Computing and Intelligent Edge for Smart City . . . . .	17
2.1.1 Edge Computing and Smart City . . . . .	17
2.1.2 Intelligent Edge in Smart City . . . . .	19
2.2 Non-intrusive load monitoring (NILM) . . . . .	20
2.2.1 Various Types of NILM Method . . . . .	21
2.2.2 Load Identification for Electric Vehicle (EV) charging load . . . . .	22
2.2.3 EV charging patterns . . . . .	23
2.2.4 Recent studies for extracting EV charging loads . . . . .	23
2.2.4.1 Training-Free approach . . . . .	24
2.2.4.2 Unsupervised Learning-based approach . . . . .	26
2.2.4.3 Deep Learning-based approach . . . . .	26
2.2.5 Limitations of Existing Edge Computing based NILM method . . . . .	27
2.3 Datasets for Load Identification . . . . .	28
2.4 Non-Intrusive Load Monitoring Toolkit (NILMTK) . . . . .	29

2.4.1	Limitations of current NILMTK and researches . . . . .	29
2.5	Summary . . . . .	29
CHAPTER 3. RESEARCH DESIGN . . . . .		30
3.1	Hypothesis . . . . .	30
3.2	Approach and System Design . . . . .	30
3.3	Dataset and Features . . . . .	31
3.4	Edge Computation Platform . . . . .	34
3.5	NILM implemenations on an edge device . . . . .	35
3.5.1	Data Converter . . . . .	35
3.5.2	NILM Algorithms . . . . .	36
3.5.2.1	Combinatorial optimization (CO) . . . . .	36
3.5.2.2	Factorial hidden Markov model (FHMM) . . . . .	36
3.5.3	EV profile extraction algorithm . . . . .	37
3.6	Performance Evaluation Criteria . . . . .	38
3.6.1	Root Mean Square Error (RMSE) . . . . .	38
3.6.2	Runtime . . . . .	38
3.7	Adoption of Cloud Server Platform . . . . .	39
3.8	The flowchart of Experiment Scenario . . . . .	39
3.9	Summary . . . . .	40
CHAPTER 4. EXPERIMENT . . . . .		41
4.1	Dataset and Features . . . . .	41
4.2	Experimental Environment . . . . .	41
4.2.1	Libraries used on Raspberry Pi . . . . .	41
4.2.2	Cloud Server Setup for Raspberry Pi . . . . .	43
4.2.3	Location . . . . .	43
4.3	Experimental Result . . . . .	44
4.3.1	Data Preprocessing . . . . .	44
4.3.2	Data Analysis Result . . . . .	47
4.3.3	NILM result for EV charging loads . . . . .	50
4.3.3.1	NILM using short-term data . . . . .	50



4.3.3.2 NILM using long-term data . . . . .	54
CHAPTER 5. SUMMARY . . . . .	66
REFERENCES . . . . .	68

## LIST OF TABLES

2.1	Summaries of the most frequently used Dataset . . . . .	28
3.1	Dataport accessible to university members with free of charge . . . . .	32
3.2	Technical Specs of Raspberry Pi 4 Model B . . . . .	34
3.3	System Requirements of OwnCloud . . . . .	39
4.1	The time profiles of EV charging load (short-term data) . . . . .	50
4.2	RMSE for the short-term data experiment . . . . .	53
4.3	The time profiles of EV charging load (long-term data) . . . . .	55
4.4	RMSE for the long-term data experiment based on <i>sum_power</i> . . . . .	57
4.5	The output of EV charging profiles based on <i>sum_power</i> . . . . .	60
4.6	RMSE for the long-term data experiment based on <i>net_power</i> . . . . .	61
4.7	The output of EV charging profiles based on <i>net_power</i> . . . . .	64

## LIST OF FIGURES

1.1	An end-to-end architecture for identifying EV load using edge computing . . . . .	13
2.1	An overview of the IoT-based energy management architecture for smart cities (Liu, Yang, Jiang, Xie, & Zhang, 2019) . . . . .	18
2.2	General Framework of NILM Approach . . . . .	20
2.3	An aggregated load data obtained by single point (Hart, 1992b) . . . . .	20
2.4	Projection of EV and Hybrid Vehicles Sales by 2050 (U.S. Energy Information Administration, 2018) . . . . .	22
2.5	Four typical EV charging profiles (Zhao, Yan, & Ren, 2019) . . . . .	23
3.1	Overview of System Design . . . . .	31
3.2	Ground-truths of residential data (2 weeks) (Pecan Street Inc, 2019a) . . . . .	32
3.3	The effect of solar influx in the electrical grid. (a) The energy amount consumed by electric vehicle tagged as ‘car1’. (b) The energy amount generated by solar panel tagged as ‘solar’. (c) The measuring power taken from or fed to the electrical grid tagged as ‘grid’. . . . .	33
3.4	An end-to-end system flow . . . . .	40
4.1	Packages used for NILMTK on Raspberry Pi . . . . .	42
4.2	Data Exporting to the cloud server of Raspberry Pi . . . . .	43
4.3	Experimental Environment and Raspberry Pi 4 B . . . . .	43
4.4	Specifying types of columns to Pandas . . . . .	44
4.5	Specifying types of columns to Pandas . . . . .	45
4.6	Making chunks of data on Raspberry Pi . . . . .	45
4.7	Appliances graph from ID=5679 . . . . .	47
4.8	A pie chart of power data from ID=5679 in August 2019 . . . . .	47
4.9	(a) 1-second, (b) 1-minute, (c) 15-minute power data from ID=5679 . . . . .	48
4.10	1-minute Power data from ID=5679. (a) ‘Site meter’= the sum of power consumption of electric appliances ( <i>sum_power</i> ), (b) ‘Site meter’= power measured on the grid - solar generated power ( <i>net_power</i> ) . . . . .	49
4.11	The short-term data experiments . . . . .	51

4.12	The ground truth and predictions of EV charging loads using CO . . . . .	53
4.13	The ground truth and predictions of EV charging loads using FHMM . . . . .	54
4.14	The long-term data experiments . . . . .	56
4.15	The ground truth and predictions of EV charging loads using CO . . . . .	58
4.16	The ground truth and predictions of EV charging loads using FHMM . . . . .	59
4.17	The ground truth and predictions of EV charging loads using CO . . . . .	62
4.18	The ground truth and predictions of EV charging loads using FHMM . . . . .	63
4.19	Runtime of CO classifier . . . . .	65
4.20	Runtime of FHMM classifier . . . . .	65

## LIST OF ABBREVIATIONS

NILM	Non-Intrusive Load Monitoring
EV	Electric Vehicle
IoT	Internet Of Things
AI	Artificial Intelligence
DRL	Deep Reinforcement Learning
SOC	State of Charging
SoC	System-on-Chip
HMM	Hidden Markov Model
LSTM	Long-Short Term Memory algorithm
kNN	k-Nearest Neighbors
SVM	Support Vector Machine
DOE	U.S. Department of Energy
REDD	Reference Energy Disaggregation Dataset
BLUED	Building-Level fULLy-labeled dataset for Electricity Disaggregation
AMPds	Almanac of Minutely Power datasets
ICA	Independent Component Analysis
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
NILMTK	NILM toolkit
CO	Combinatorial optimization
FHMM	Factorial hidden Markov model (FHMM)

## **GLOSSARY**

Edge Computing – one of the computing paradigms that includes data storage and computing being performed at the “edge” within the device.

Internet of Things – an extension of the internet in which physical devices, vehicles, buildings and other physical items are enabled to collect and exchange data and provide services via connections.

Non-Intrusive Load Monitoring – methods that are beneficial when identifying different types of electric loads based on unique patterns of electrical appliances.

Smart Grid – an electrical grid that includes various operation and energy measures including smart meters, smart appliances, renewable energy resources, etc.

## ABSTRACT

The accelerated urbanization of countries has led the adoption of the smart power grid with an explosion in high power usage. The emergence of Non-intrusive load monitoring (NILM), also referred to as Energy Disaggregation has followed the recent worldwide adoption of smart meters in smart grids. NILM is a convenient process to analyze composite electrical energy load and determine electrical energy consumption.

A number of state-of-the-art NILM (energy disaggregation) algorithms have been proposed recently to detect various individual appliances from one aggregated signal observation. Different kinds of classification methods such as Hidden Markov Model (HMM), Support Vector Method (SVM), neural networks, fuzzy logic, Naive Bayes, k-Nearest Neighbors (kNN), and many other hybrid approaches have been used to classify the estimated power consumption of electrical appliances from extracted appliances signatures. This study proposes an end-to-end edge computing system with an NILM algorithm, which especially focuses on recognizing Electric Vehicle (EV) charging. This system consists of three main components: (1) Data acquisition and Preprocessing, (2) Extraction of EV charging load via an NILM algorithm (Load identification) on the NILMTK Framework, (3) and Result report to the cloud server platform.

The monitoring of energy consumption through the proposed system is remarkably beneficial for demand response and energy efficiency. It helps to improve the understanding and prediction of power grid stress as well as enhance grid system reliability and resilience of the power grid. Furthermore, it is highly advantageous for the integration of more renewable energies that are under rapid development. As a result, countless potential NILM use-cases are expected from monitoring and identifying energy consumption in a power grid. It would enable smarter power consumption plans for residents as well as more flexible power grid management for electric utility companies, such as Duke Energy and ComEd.

## CHAPTER 1. INTRODUCTION

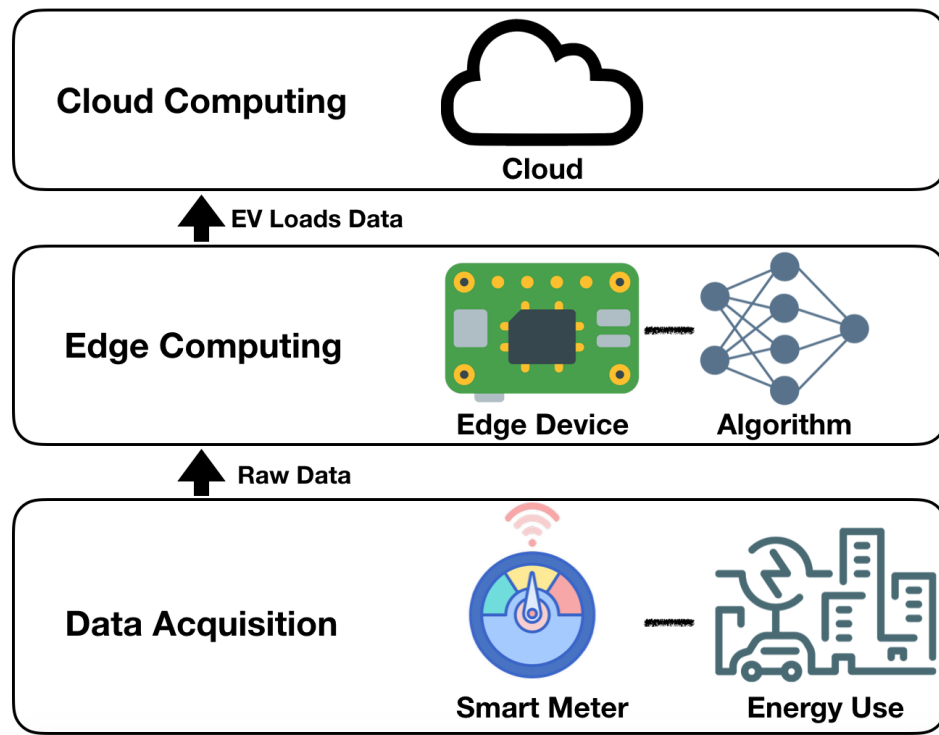
This thesis focuses on edge computing for smart grids, which aims to recognize electric vehicle charging loads. This chapter provides an overview of the research study by presenting a problem statement, a research question, research significance, assumptions, limitations, and delimitations that define the extent of the study.

### 1.1 Statement of Problem

The rapid advancement of new technologies as well as accelerated urbanization of countries has led the emergence of the edge computing in smart power grids. A myriad of potential application areas of adopting edge computing in smart grids have attracted extensive research efforts for a more stable and reliable power grid from industries and academia. Decentralized power management (Chen et al., 2019) and resource allocation (Yang, Chen, Liu, Zhong, & Xie, 2019) based on data supported by IoT sensors are keys to achieving reliability and flexibility in smart power systems. Among various applications of data analysis in smart grids, the identification of energy use can be an initial step for smart grids, followed by load forecasting, failure prediction, and self-recovery. Non-Intrusive Load Monitoring (NILM) methods are beneficial when identifying different types of electric loads based on unique patterns of electrical appliances. In this study, we are interested in extracting electric vehicle (EV) charging loads by using a low-cost edge device. According to the U.S. Energy Information Administration (EIA) (U.S. Energy Information Administration, 2018), there will be over 2.3 million new light-duty EVs and hybrid by 2050. Accordingly, there have been several approaches, such as supervised learning, unsupervised learning, and training-free algorithm for recognizing EV charging load as the impact of EV keeps growing. Started with (Zhang et al., 2014), a majority of algorithms (Munshi & Mohamed, 2019; Wang, Du, Ye, & Zhao, 2018; Zhao, Yan, & Ma, 2019) for extracting EV charging loads evaluated the efficiency of their algorithms using Pecan Street database (Pecan Street Inc, 2019a), one of the most cited and easily accessible datasets.



Currently, there are a few pieces of research for applying machine learning models on an edge device to identify electric power loads. (Lai, Chien, Yang, & Qiang, 2019) adopts Long-Short Term Memory (LSTM) algorithm to recognize industrial electrical equipment. The downside of this study is that an edge device is used only to preprocess raw data. The preprocessed data is fed into a cloud layer for running the complicated algorithm. The research (Sirojan, Phung, & Ambikairajah, 2017) runs a neural network algorithm by using an embedded device (myRIO-1900) to identify four small electrical appliances: a fan, fluorescent, laptop, and incandescent. However, the cost of the embedded device used is over \$1,000.



*Figure 1.1.* An end-to-end architecture for identifying EV load using edge computing

To address the above limitations, this study proposes an implementable end-to-end solution that enables low-cost small edge device to process a massive amount of smart meter data in order to recognize and identify a substantial electric load, specifically electric vehicle (EV) charging loads. The concept of cloud computing is adopted in a way that essential results are sent to the cloud server. The cloud server in this study aims to adapt and extend edge computing to be easily integrated with other technologies.

After all, the final output provided by this study is an end-to-end system by making use of NILM focusing on EV charging loads, proceeded by processing data on a cost-effective embedded device and as shown in Figure 1.1.

## 1.2 Significance

Recent years have witnessed active research in building smart cities using edge computing (Chen et al., 2019; Khan et al., 2019; Liu et al., 2019). In particular, the experimental works in real-life scenarios based on the intelligent edge were conducted in (Ferrandez, Mora, Jimeno-Morenilla, & Volckaert, 2018; Lai et al., 2019; Sirojan et al., 2017; Syafrudin, Fitriyani, Alfian, & Rhee, 2019). Also, many works (Munshi & Mohamed, 2019; Wang et al., 2018; Zhao et al., 2019) have focused on identifying EV charging loads to grid operators develop efficient and effective management strategies and strengthen grid resilience by minimizing the consequences of sudden massive changes in a grid. Taking inspiration from recent research, this study is intended to verify whether a low-power edge device is able to recognize EV charging loads in a smart grid as part of active research areas in developing a smart city. This research adopts Non-Intrusive Load Monitoring (NILM) algorithms and discovers the advantages and limitations of edge computing. Eventually, the study is expected to contribute to (1) extending the knowledge of capabilities and potentialities of edge computing (2) increasing the resilience of smart power grids.

## 1.3 Research Question

This study focuses on answering the following questions.

- Can a low-power edge device identify the electric vehicle (EV) charging load from smart meter datasets using Non-Intrusive Load Monitoring (NILM) algorithms, and further send results to the cloud?
- Is it possible to recognize EV charging loads with the consideration of solar-generated power with the emerging supply PV systems?

- Which sampling frequency is the most appropriate to recognize EV charging loads in the low-power edge device?

#### 1.4 Assumptions

The assumptions for this study include:

- An edge device is used to load, analyze, and process data, which is already collected by real-world smart meters.
- Although there are other factors that can affect the quality of communication between an edge device and cloud platform, this study doesn't consider any other disturbances of communication such as network attack.

#### 1.5 Limitations

This study is performed acknowledging the following constraints:

- The performance of NILM methods powered by edge computing can be affected by various factors, such as the capacity of edge devices and network.
- This study investigated different kinds of existing NILM methods from recent prior studies, which have shown good performance.

#### 1.6 Delimitations

The delimitations for this study include:

- This study is not intended to directly collect power data from smart meters due to safety issues, and therefore, based on real-world datasets that include aggregated power load signals and the ground truth of EVs (Pecan Street Inc, 2019a).

- This research tried to find the most appropriate NILM methods based on machine learning algorithms for identifying EV charging loads in power grids. However, the research does not compare all different kinds of NILM methods since recognizing EV charging loads is one of the newest tasks in NILM, and there are not many algorithms targeting EV charging loads (Murugan, Garg, & Singh, 2017).

### 1.7 Summary

This chapter provided an overall description of the research that contains a statement of problem, significance, research question, limitations, and delimitations that are considered in this study.

## **CHAPTER 2. REVIEW OF LITERATURE**

This chapter provides a review of the literature relevant to this study. This chapter is organized as follows. Section 2.1 describes evolving edge computing and Intelligent Edge for Smart City. Section 2.2 Non-intrusive load monitoring (NILM), especially for recognizing Electric Vehicle (EV) charging and machine learning algorithms. Section 2.3 discusses data sets for an EV charging recognition system evaluation. At last, a unique toolkit for NILM (NILMTK) is introduced in Section 2.4.

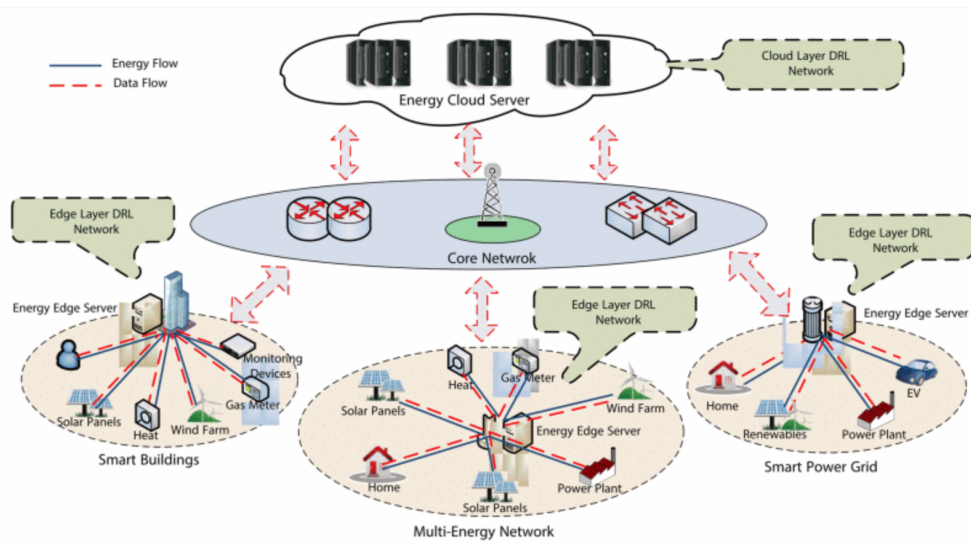
### 2.1 Edge Computing and Intelligent Edge for Smart City

Edge computing is one of the computing paradigms that includes data storage and computing being performed at the “edge” within the device. Data is analyzed and processed locally, such that only significant events are transmitted to the cloud layer. The advent of this concept stems from trials to overcome the limitations of cloud computing. Managing a network, maintaining connectivity, and providing services in a cloud environment is not easy, especially when it comes to Internet of Things (IoT). The benefits of edge computing are countless. Edge computing can handle acquisition, pre-processing, and manipulating big data without the high cost of network bandwidth and high delay or latency, which are some of the drawbacks of cloud computing. Also, processing private data at the edge helps to prevent privacy or security issues.

#### 2.1.1 Edge Computing and Smart City

Among various application scenarios that will be beneficial from the concept of Edge Computing (Yi, Li, & Li, 2015), one of the application scenarios is ‘Mobile Big Data Analytics’ (Bonomi, Milito, Zhu, & Addepalli, 2012). The study describes that the data processing in the edge device will be the crucial technique to tackle analytics on a large scale of data in IoT.

Aided by capabilities of edge computing, there has been recent research on using edge computing to build smart cities (Chen et al., 2019; Khan et al., 2019; Liu et al., 2019). (Liu et al., 2019) introduces IoT-based energy management architecture for smart cities, as shown in Figure 2.1. The study highlights the use of the architecture in Smart Buildings, Multi-Energy Network, and Smart Power Grid. (Khan et al., 2019) introduces scenarios of an edge computing enabled smart city that includes Smart Transportation System, Smart Buildings, Smart Grid, etc. (Chen et al., 2019) also introduces an architecture of edge computing for IoT-based smart grids for micro-grid systems, metering systems, and surveillance systems. They all describe the possible contributions of edge computing in the smart power grid and emphasize the need for intelligent edge. The concept of intelligent edge and the smart power grid will be discussed in the following subsections.



*Figure 2.1.* An overview of the IoT-based energy management architecture for smart cities (Liu et al., 2019)

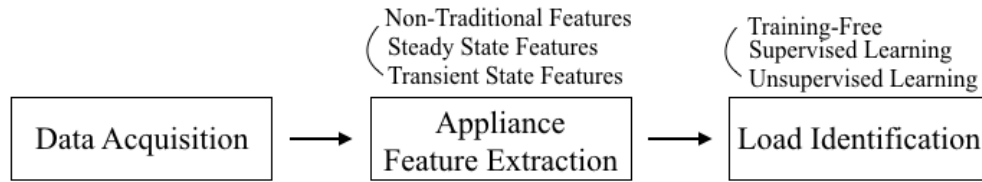
### 2.1.2 Intelligent Edge in Smart City

It is no doubt that there are always trade-offs when it comes to taking advantage of the features of new concepts. Likewise, there are various challenges and issues that have to be considered and solved to make the most of the benefits of edge computing. (Khan et al., 2019) points out that one of the research challenges on realization for edge computing is how to separate useful data sets from noisy data in a massive amount of data generated while edge processing. The use of Artificial Intelligence (AI) at the edge can be key to solving this challenge. Filtering out noisy data, processing/analyzing real-time data, and predicting future trends can be good examples of using AI at the edge. When the data is acquired, saved, and processed with machine learning algorithms at the network edge (e.g., embedded edge devices), it is specifically referred to as “edge intelligence” (Plastiras, Terzi, Kyrkou, & Theocharides, 2018).

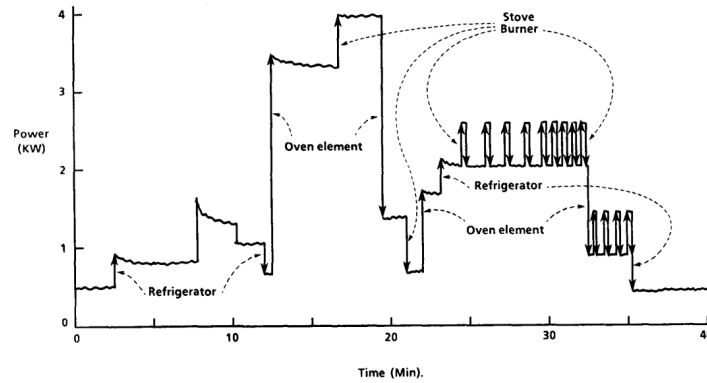
The need for intelligent edge was emphasized in numerous studies that consider using edge computing in building smart cities. (Liu et al., 2019) discusses the deployment of edge computing with deep reinforcement learning (DRL). (Chen et al., 2019) focuses on solving rapid response for user’s requirements, intelligent scheduling, intelligent maintenances based on their architecture. The limitation of such research is in the gap between the design of architecture and an actual implementation with an “edge”. They focus on introducing conceptual designs of their architectures rather than a practical experiment, which can describe the feasibility of intelligent edge. (Liu et al., 2019) adds four layers, which include the application layer, cognition layer, network layer, and sensing layer. Likewise, (Chen et al., 2019) introduces the cloud - application layer, data layer, network layer, and the device layer. In the meantime, two studies are presenting practically implementable end-to-end solutions for utilizing edge computing in each scenario for a smart city. (Ferrandez et al., 2018) separated concepts of edge and fog computing. Edge computing is for sensing while fog computing is for running a machine learning algorithm on a Raspberry Pi. Likewise, (Syafudin et al., 2019) conducted a practical experiment to manage renewable power by adopting pattern recognition and decision trees methods on a Raspberry Pi based on data collected by different kinds of sensors.

## 2.2 Non-intrusive load monitoring (NILM)

NILM is a technique to analyze energy consumption to identify what appliances are used in a power grid, first introduction by (Hart, 1992b) in the late 1980's. NILM is also known as energy disaggregation, which aims to separate individual appliance energy consumption from aggregated electricity consumption data. The general framework for electric load identification contains event detection, feature extraction, and load identification using the extracted features, as shown in Figure 2.2. Figure 2.3 represents an example of an aggregated load data to be identified.



*Figure 2.2. General Framework of NILM Approach*



*Figure 2.3. An aggregated load data obtained by single point (Hart, 1992b)*



### 2.2.1 Various Types of NILM Method

The algorithm presented by Hart (Hart, 1992b) intends to match changes between sharp edges in the aggregated energy and information of states' shifts of different kinds of appliances in signature database. The signature database is manually labeled with different traits of electrical appliances and no machine learning methods are used. A large number of researchers have paid attention to NILM, and comprehensive reviews on NILM can be found in (Zoha, Gluhak, Imran, & Rajasegarar, 2012).

The majority of research on NILM method focuses on supervised machine learning models. The supervised learning requires labeled data sets for training the classifier (Zoha et al., 2012). The typical supervised learning tasks are classification (e.g., spam filter) and regression (e.g., car price prediction) (Gron, 2017). Many different state-of-the-art NILM algorithms have been proposed recently to detect various individual appliances from one aggregated signal observation. A wide range of classification methods, such as Hidden Markov Model (HMM), Support Vector Method (SVM), neural networks, fuzzy logic, Naive Bayes, k-Nearest Neighbors (kNN) have been presented for NILM. The supervised machine learning mechanism is appropriate to approach for NILM since the model would be able to recognize and identify operations of electrical appliances from the aggregated load measurement based on measurements of power and current.

Unlike most of the supervised approaches that rely on event detection for classification, the unsupervised methods are non-labeled, which means non-event-based. Some of the unsupervised learning tasks are clustering, visualization and dimensionality reduction, and association rule learning. The unsupervised learning techniques attempt to disaggregate the collected loads directly, without the need of event detection (Zoha et al., 2012). As energy consumption patterns are getting more complicated, many other hybrid approaches have been used to classify the estimated power consumption of electrical appliances from extracted appliances signatures.

### 2.2.2 Load Identification for Electric Vehicle (EV) charging load

NILM has followed the recent worldwide adoption of smart grids. A smart grid is a cutting-edge technology for the transformation of the traditional power grids to reduce utility costs and environmental issues by adopting renewable energy resources, smart meters, and smart appliances. Electric Vehicle (EV) is one of the enormous power loads in a smart grid that increases the operational burden on power grids. According to the U.S. Department of Energy (DOE) (U.S. Energy Information Administration, 2018), there will be over 2.3 million new light-duty EVs and hybrid by 2050, as shown in Figure 2.4, and the impact of EV will be increased accordingly.

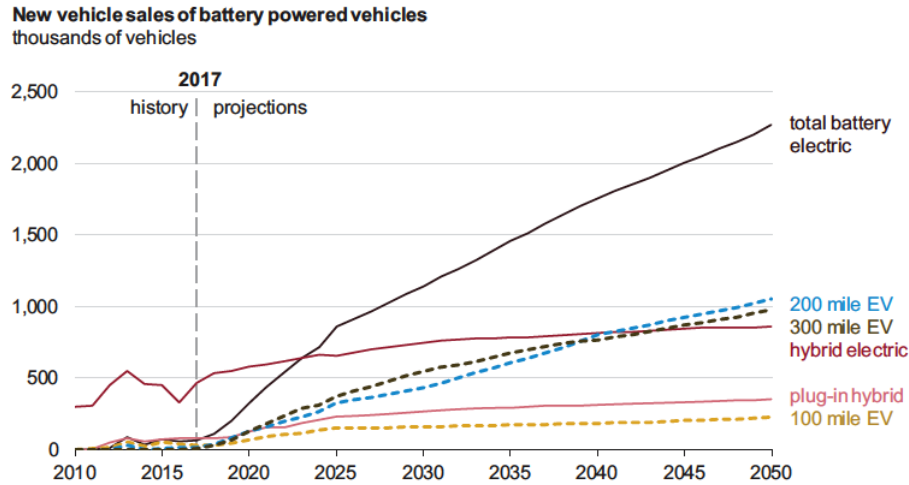


Figure 2.4. Projection of EV and Hybrid Vehicles Sales by 2050 (U.S. Energy Information Administration, 2018)

Therefore, extracting EV charging loads from aggregated energy loads is an important aspect that enables smart grid operators to keep updated and make intelligent decisions about conserving power and strengthening electrical grid resilience by balancing supply and load. In the following sections, EV charging load profiles are investigated along with discussions of recent studies on extracting EV charging loads.

### 2.2.3 EV charging patterns

In order to separate the EV charging load from aggregated power loads, it is necessary to fully understand the characteristics of EV charging load. Therefore, four typical EV charging patterns were analysed in (Zhao et al., 2019) based on datasets obtained from (Pecan Street Inc, 2019a). The EV receives power the most during Phase 1, of which charging power is almost constant, as shown in Figure 2.5. When the state of charging (SOC) of the EV battery reaches a pre-defined value, it turns to Phase 2. During Phase 2, the power level of the vehicle's battery is renewed with a different power pattern. Phase 3 is defined as a pulse maintenance stage, aiming to compensate for the loss coming from battery self-discharge. The EV's charging profile does not always have Phases 2 and 3, and the shape of the profile can be varied depending on the battery type as well as charging algorithm.

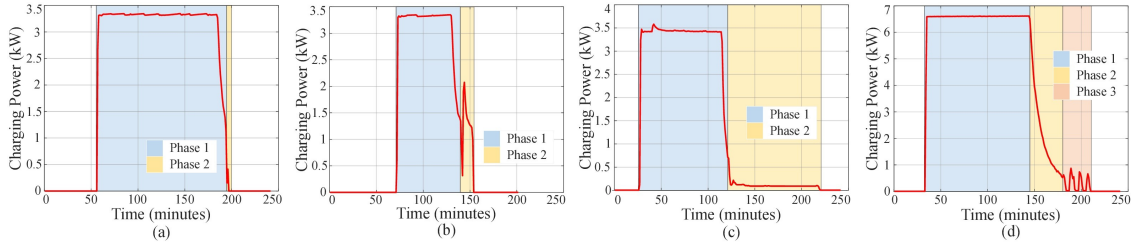


Figure 2.5. Four typical EV charging profiles (Zhao et al., 2019)

### 2.2.4 Recent studies for extracting EV charging loads

Despite the fact that a variety of NILM algorithms were presented for energy disaggregation, few NILM algorithms specially designed for identifying EV charging loads are developed. Started with (Zhang et al., 2014) in 2014, various modern and improved NILM methods have been presented based on training-free approach, unsupervised learning-based approach, and deep learning-based approach for recognizing EV charging load. (Munshi & Mohamed, 2019; Wang et al., 2018; Zhang et al., 2014; Zhao et al., 2019) are up-to-date researches that show good performance among similar studies in extracting EV charging loads from the aggregated power signals of households.

#### 2.2.4.1 Training-Free approach

One of the benefits of a non-training algorithm is that this method demands a light computational power. (Zhang et al., 2014) proposed a NILM algorithms without training process for identifying EV charging load. Overall, the proposed algorithm outperforms in disaggregation accuracy compared to the hidden Markov (HMM) algorithm. The authors especially focus on effectively mitigating interference caused by air-conditioner power signals during summer. The study states that an EV waveform with satisfactory accuracy can be reconstructed since an EV waveform height is generally ranging from 3 kW to 4 kW in most cases while the height of an AC lump is generally smaller than 3 kW.

The process used by (Zhang et al., 2014) is broken into five steps: Step 1) Thresholding the aggregated power signals, 2) Filtering particular spike-trains, Step 3) Removing residual noise using local noise amplitude, Step 4) Classifying the type (type 0, type 1, and type 2) of each segment, and Step 5) Disaggregating Energy. It is worth noting that each segment is classified into one of three types in Step 4. Type 0 can be seen as a dryer/oven waveform, an EV waveform fully overlapping with a dryer/oven waveform. Type 1 can be considered as an EV waveform, an AC waveform, or an EV waveform overlapping with other appliances. Finally, Type 2 is an EV waveform overlapping with an AC waveform. The result of the proposed algorithm shows only 7.5% for the averaged estimation error (Err) of monthly energy consumption, while the result of the HMM algorithm shows 55.6%. More details are analyzed and written in pseudo-codes in Algorithm 2.1.

Of course, there is a multiple numbers of challenges of disaggregating an EV charging load from aggregated power signals. For example, the distinguishing the AC signals from EV charging load signals can be very difficult, especially in the case where other kinds of appliances are operating and causing fluctuating residual noise. In addition, the aggregated data were sampled at 1/60 Hz. This low sampling makes some other appliance signatures available from high sampling rate no longer exist and constraints accurate pattern recognition. Finally, even though training-free approach does not require training sets, the algorithm may poorly perform for various houses since the ground-truth of EV charging load patterns should not be identical across different houses.

---

**Algorithm 2.1** Training-Free EV charging Load Extraction.

Raw Algorithm Source: (Zhang et al., 2014)

---

```
1: function EXTRACT_EV(aggregated signal  $x(t)$ , a threshold  $T_{low}$ )
2:   // Thresholding the Aggregated Signal
3:   if  $x(t) < T_{low}$  then  $\underline{x}(t) = 0$  else  $\underline{x}(t) = x(t)$ 
4:   //Filtering the Spike-Train
5:   Find segmentss with duration shorter than  $T_{seed} = 20$ , called seeds
6:   for seed in seeds do
7:     Searches the nearest segment forwardly
8:      $D_{cur} = DurationOfTheCurrentSeed$ ;  $\Gamma = 1.2$ 
9:     if  $D \triangleq (1 + \Gamma)D_{cur}$  and  $|seed - NearestSegment| < 3D_{cur}$  then
10:      Label NearestSegment as ‘spikes to remove’;
11:       $seed = nearestSegment$ 
12:     end if
13:   end for
14:   Same procedure done for segments backwardly
15:   remove all segments labeled as ‘spikes to remove’ from  $\underline{x}(t)$ 
16:   //Removing Residual Noise
17:   for segment in segments do
18:      $N_b = \min(segments[segmentindex - 5 : segmentindex])$ 
19:      $N_a = \min(segments[segmentindex : segmentindex + 5])$ 
20:      $LocalNoiseAmplitude = \text{avg}(N_b, N_a)$ 
21:     Obtain residual noise removal by  $segment - LocalNoiseAmplitude$ 
22:   end for
23:   //Classifying the Type of Each Segment
24:   for segment in segments do
25:      $peak_{nums} = \text{findpeaks}(segment)$ 
26:      $segment.type = peak_{nums}$ 
27:   end for
28:   //Energy Disaggregation
29:    $eff\_Width$  =the width of a segment at bottom
30:    $eff\_Height$  =the height at which  $0.8 * \text{segment's width}$ 
31:   for segment in segments do
32:      $//segment.type == 0$ 
33:     if  $eff\_Height < 5.5kW$  then  $segment = \text{dryer/oven}$  else  $segment = EV$ 
34:      $//segment.type == 1$ 
35:     if  $eff\_Width > 250min$  then  $segment = AC$  else  $segment = EV$ 
36:      $//segment.type == 2$ 
37:     Distinguish if EV is upper or bottom parts of the segment
38:   end for
39:   return Obtained the final extracted EV charging Loads
40: end function
```

---

#### 2.2.4.2 Unsupervised Learning-based approach

According to (Munshi & Mohamed, 2019), EV charging load patterns are composed of three stages: Stage 1) gradual increase in charging load, Stage 2) steady-state charging load, and Stage 3) gradual decrease in charging load. These stages are understood with the same context of what Figure 2.5 describes. The authors point out that the previous training-free approach (Zhang et al., 2014) only works for extracting the stage 2 of EV charging loads and neglects charging sessions less than 30 minutes. The unsupervised learning-based approach identifies Stage 2 of EV charging by applying Independent Component Analysis (ICA) method, extracting EV charging vector, removing false positive (FP) extractions, and estimating the amplitude of the extracted EV charging loads. Eventually, the algorithm extracts Stage 1 as well as Stage 3 after extracting Stage 2. The proposed algorithm improved overall performance in F-score, as well as accuracy compared to the training-free algorithm (Zhang et al., 2014). The error rate of actual data and extracted results has gone down to 2.66%. More details are analyzed and written in pseudo-codes in Algorithm 2.2.

#### 2.2.4.3 Deep Learning-based approach

One of the advantages of a deep learning-based approach is that it does not require hand-crafted features compared to supervised learning-based approaches. (Wang et al., 2018) attempts to identify starting time, charging periods, and initial SOC of EV charging profiles using deep learning. The deep learning framework proposed in this study consists of multilayer perceptrons (MLPs), the undercomplete autoencoder, the denoising autoencoder, and 1D Convolutional Neural Network(CNN). The denoising autoencoder is sensitive to its input demands the data pre-processing phase. The pre-processing includes finding the effective interval for target appliance signal, selecting slide window size, filtering based on the threshold of amplitude, and normalizing labels. This algorithm adopted a mini-batch training approach and an entire batch approach to compare CPU running time. Finally, an average accuracy of the mini-batch approach and the entire-batch approach for the starting and end time were 0.009124 and 0.08663 respectively, and mean square reconstruction loss was 0.03495.

---

**Algorithm 2.2** Unsupervised Learning-based EV charging Load Extraction.

Raw Algorithm Source: (Munshi &amp; Mohamed, 2019)

---

```
1: function EXTRACT_EV(data  $\mathbf{x}$ ,  $N$ ,  $M$ ,  $p_{max}$ )
2:    $m = 0$ ; templates  $\mathbf{S}$  // Initialization
3:   while  $M$  not  $m$  do // Iterative Process
4:      $m = m + 1$ ;  $p = 0$ 
5:     Apply ICA for actual load  $\mathbf{x}$ , and template  $s_m$ 
6:     Obtain the extracted EV charging loads matrix  $\mathbf{Z}$ 
7:     while  $p_{max}$  not  $p$  do
8:        $p = p + 1$ 
9:       Obtain the extracted EV charging load solutions  $c_p$ 
10:      Detect and remove false extractions  $\mathbf{FPs}$ 
11:      Update the extracted EV charging loads  $c_p$ 
12:      Construct  $EST_x$  of  $c_p$  by one of the estimation methods 1 4
13:      Compute  $Error_{pm}$ 
14:    end while
15:    Compute  $Error_{1m}$ ; Obtain best extracted EV charging load  $\tilde{c}_m$ 
16:  end while
17:  //Local Estimation & Final EV charging Load Result
18:  Obtain  $\tilde{c}_b$  and EV charging loads that exist in the same category
19:  Construct  $EST_{2x}$  to improve the amplitude estimation by estimation Method 1 4
20:  Compute  $Error_{2b}$ 
21:  Obtain the final extracted EV charging Load  $\tilde{\mathbf{F}}$ 
22:  return  $\tilde{\mathbf{F}}$ 
23: end function
```

---

### 2.2.5 Limitations of Existing Edge Computing based NILM method

With the help of edge computing paradigm, smart grids based on IoT devices will provide the real-time analysis and processing of massive data, foster efficient power management for stability, and eventually realize the digitalization of smart grids (Okay & Ozdemir, 2016). As previously mentioned, a couple of architectures and frameworks that possibly develop a smart city supported by edge computing have been introduced and proposed in recent studies. Furthermore, (Lai et al., 2019; Sirojan et al., 2017) designed and implemented machine learning models for load identification based on edge devices. (Lai et al., 2019) used edge device to pre-process data sets and send the pre-processed data up to the cloud framework for running Long-Short Term Memory (LSTM) algorithm.

On the contrary, (Sirojan et al., 2017) used an embedded device named myRIO-1900 to preprocess raw data as well as running a neural network algorithm to identify four different electrical appliances. Note that the two studies contributed to increasing the practical usefulness of edge computing based on real-world data, not limited to proposing abstract architecture or framework. One downside of (Sirojan et al., 2017) is that the cost of the embedded device used is over \$1,000 and still very expensive compared to Raspberry Pi or Odroid. Additionally, none of the former studies is presenting a practically implementable end-to-end solution for EV charging load monitoring and power estimation using edge devices.

### 2.3 Datasets for Load Identification

In order to implement and deploy an energy disaggregation algorithm, most studies utilize data from previously collected and provided on the purpose of research. Table 2.1 presents four datasets which are readily accessible and most cited from countless NILM researches: the Reference Energy Disaggregation Dataset (REDD) (Kolter & Johnson, 2011), the Building-Level fully-labeled dataset for Electricity Disaggregation (BLUED) (Anderson et al., 2012), the Pecan Street Database (Pecan Street Inc, 2019a), and the Almanac of Minutely Power dataset (AMPds) (Makonin, Ellert, Bajic, & Popowich, 2016). These open public data are invaluable as we can compare NILM algorithms with other existing results of recent studies. Despite the increasing demand of electric vehicles, not many publicly available NILM datasets have EV charging load data other than Dataport.

Table 2.1. *Summaries of the most frequently used Dataset*

	Pecan Street	REDD	BLUED	AMPds
Institution	Pecan Street Inc.	MIT	CMU	Simon Fraser Univ.
Location	TX/CA/NY, USA	MA, USA	PA, USA	BC, Canada
Data frequency (Hz)	1	1	60	1/60
Duration (days)	365	33.5	7.1	365
Households	1391	6	1	1



## 2.4 Non-Intrusive Load Monitoring Toolkit (NILMTK)

NILMTK is an open source toolkit that makes it more convenient for researchers to conduct a comparative analysis of NILM algorithms across a set of existing datasets (N. Batra, 2015; N. Batra et al., 2014; Kelly et al., 2014). The platform is important because previous researches had difficulties in comparing algorithms utilizing more than one kind of datasets with accuracy metrics. A number of researchers have paid attention to this framework to address such challenges in NILM researches (Buneeva & Reinhardt, 2017; Dinesh et al., 2017; Kelly & Knottenbelt, 2015a; Osathanunkul & Osathanunkul, 2019).

### 2.4.1 Limitations of current NILMTK and researches

NILMTK is a unique tool that allows comparative NILM research across different datasets. However, there are some limitations when utilizing NILMTK for recent studies. First, There have not been NILM researches based on Dataport in NILMTK format. The Dataport in NILMTK format was first introduced in (Parson et al., 2015), but this is not applicable for recent studies due to new data access policies of Pecan Street Inc. The most recent study using Dataport in NILMTK format is (Gaur, Makonin, Bajić, & Majumdar, 2019), but this study is only focused on the ground-truth of event anomalies. Second, the vast majority of existing researches have utilized REDD datasets. It is assumed that the current data converter of NILMTK is not compatible with Dataport. Third, the solar power influx cannot be considered with the current NILMTK due to its limitations of software architecture(Dinesh et al., 2017).

## 2.5 Summary

This chapter provided a review of the literature relevant to edge computing for smart grid and NILM methods for EV charging load. The next chapter presents the research methodology of this study.

## CHAPTER 3. RESEARCH DESIGN

The goal of this research is to build an electric vehicle (EV) charging load identification system from smart meter datasets with the combination of edge computing and Non-Intrusive Load Monitoring (NILM) algorithms. This chapter describes the system overview, followed by edge computation platform, NILM algorithms, and cloud platform.

### 3.1 Hypothesis

The hypothesis of this research is as follows:

- $H_0$ : Using NILM algorithms, an edge device can extract EV charging load patterns with a success rate greater than 90%.
- $H_1$ : Using NILM algorithms, an edge device cannot extract EV charging load patterns with a success rate greater than 90%.

### 3.2 Approach and System Design

The overall design for the proposed system is divided into three parts, as shown from the bottom to the top in Figure 3.1. The first part is data acquisition phase. The datasets from Pecan Street Inc. (Pecan Street Inc, 2019a), based on real-world datasets, including aggregated power load signals and the ground truth of electric vehicle charging loads, are input to an edge device. The next part is the edge computing phase, which consists of an edge device and NILM algorithm, which enable gathering and processing data from the data acquisition phase. The promising edge device used for this study is Raspberry Pi 4 Model B. The NILM model embedded in the edge device pre-processes the data and recognize electric vehicle charging loads out of aggregated power loads. The performance of algorithms is measured by several metrics in order to evaluate the effectiveness of the algorithms. After the edge computing phase, the results from the algorithms are stored and then eventually forwarded to the cloud via wireless communication.

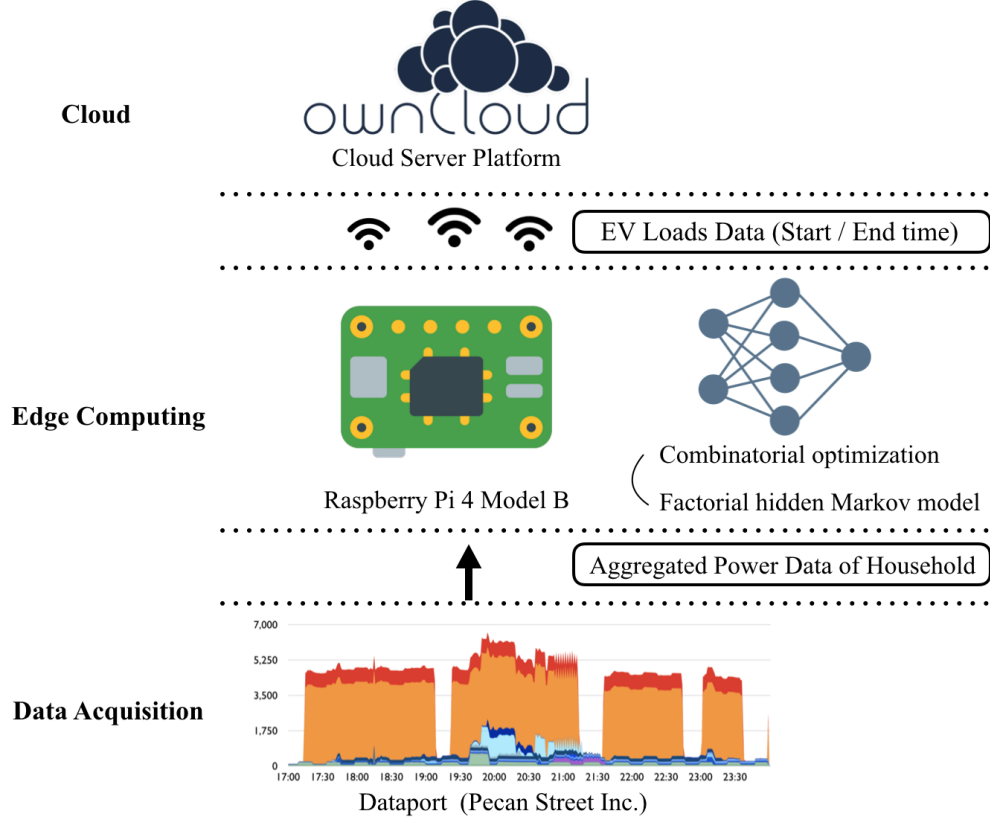


Figure 3.1. Overview of System Design

### 3.3 Dataset and Features

As described in Table 2.1, there are four datasets typically used for NILM studies. The datasets used for this study are Dataport database (Pecan Street Inc, 2019a) obtained from Pecan Street Inc (Pecan Street Inc, 2019b). Of course, some limitations have shown up during the research using datasets from Pecan Street Inc.

Dataport database used to provide electric power data collected from 1391 households in Texas, Colorado, California, etc. recorded every 1 minute. Now, they have changed data sharing policies as of January 2020. University members are able to access to 1-second, 1-minute, and 15-minute energy datasets from three regions, which are New York, California, and Austin, as shown in Table 3.1.

Table 3.1. *Dataport accessible to university members with free of charge*

	New York	California	Austin
1-second	35GB	N/A	36GB
	12GB		30GB
	11GB		35GB
	11GB		35GB
1-minute	1.2GB	1.8GB	2.6GB
15-minute	74MB	139MB	173MB

Nonetheless, there are many advantages of using datasets from Pecan Street Inc. compared to other datasets are as follows:

- Dataport contains unique datasets in that it ensures the amount of EV charging loads, which other existing datasets do not have.
- The ground-truth power signals of up to seventy appliances, such as EV, AC, refrigerator, microwave, furnace, dryer, oven, dishwasher, cloth-washer, bedroom-lighting, and bathroom-lighting are available in the database. Figure 3.2 displays ground-truths of residential data collected from house ID: 114.

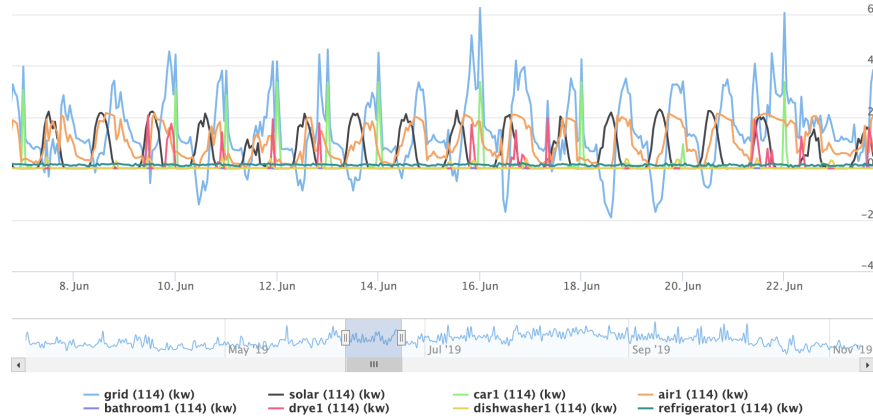
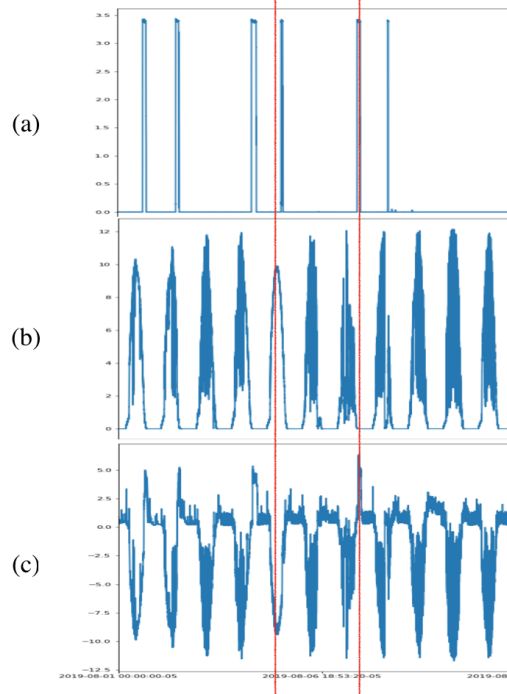


Figure 3.2. Ground-truths of residential data (2 weeks) (Pecan Street Inc, 2019a)

- Each house data contain aggregated power signals as well as each power profile for six months (e.g., May to October 2019 from the New York region). This allows us to watch the trends of EV charging patterns across the seasons.
- Most NILM algorithms for extracting EV charging data from aggregated load data used Dataport database. Therefore, the database is appropriate to test and compare algorithms' performance in practice.

The focus of this research is to recognize and identify EV charging data from a power grid. As more and more houses install photovoltaic (PV) systems, the amount of generated energy from PV significantly affects the power measured on the power grid, as presented in Figure 3.3.



*Figure 3.3.* The effect of solar influx in the electrical grid. (a) The energy amount consumed by electric vehicle tagged as 'car1'. (b) The energy amount generated by solar panel tagged as 'solar'. (c) The measuring power taken from or fed to the electrical grid tagged as 'grid'.

Given the background data and knowledge, none of the previous researches considered generated energy by the solar PV system, especially when it comes to disaggregating EV charging load in Dataport. As the effect of PV energy is significant for the power on the electrical grid, this also should be considered when extracting EV charging loads from the grid.

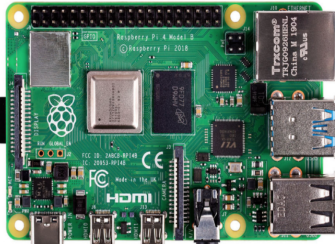
### 3.4 Edge Computation Platform

Though there exist a variety of single-board computers, Raspberry Pi and Odroid are among the most popular due to their low cost, reasonable performance, and support availability. Although Odroid has an outstanding computing power in terms of small computers, Raspberry Pi 4 Model B is selected in this research because:

- Raspberry Pi 4 Model B has a vast community behind it, especially in academia, developing and maintaining software compared to any other edge devices.
- Raspberry Pi 4 Model B is easier to use with regards to networking. 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2 can be used in Raspberry Pi while Odroid XU4 lacks built-in wireless communication functions, which requires an extra Wi-Fi module to get connected with a cloud server platform.
- Raspberry Pi 4 Model B has an advantage with pricing, as the 4GB model is \$55, which is cheaper than the Odroid XU4 with extra modules.

More details of Raspberry Pi 4 Model B are as follows:

Table 3.2. *Technical Specs of Raspberry Pi 4 Model B*

Raspberry Pi 4 Model B	Technical Specs
	<ul style="list-style-type: none"><li>• CPU: Broadcom Quad core Cortex-A72</li><li>• RAM: 4G DDR4</li><li>• GPU: 500 MHz VideoCore VI</li><li>• USB ports: 2x USB 3.0/2x USB 2.0</li><li>• HDMi ports: Dual Micro HDMI ports</li><li>• Wireless: 802.11 ac (2.4/5 GHz), Bluetooth 5.0</li><li>• OS: Debian Linux 10 based</li></ul>

### 3.5 NILM implemenations on an edge device

As introduced in Section 2.4, Non-Intrusive Load Monitoring Toolkit (NILMTK) allows easier importing of datasets and comparing between algorithms (N. Batra et al., 2014). The aims of NILMTK include the followings: First, NILMTK provides a standard data structure (NILMTK-DF) for energy datasets. Second, NILMTK presents analytical and diagnostic functions for better understandings of datasets. Lastly, NILMTK offers a couple of NILM algorithms that make it possible for researchers to compare the performance of existing or new NILM algorithms.

#### 3.5.1 Data Converter

With the help of general toolkits for machine learning tasks, NILMTK extends the capacities of such toolkits as an energy disaggregation toolkit. NILMTK is implemented in Python, which is one of the most flexible and often used languages supporting machine learning research along with libraries such as Scikit-learn (Pedregosa et al., 2011), Numpy (Oliphant, 2006), and Pandas (McKinney et al., 2010).

As of March 2020, NILMTK provides data converters (P. Batra Kelly, 2019b), including the following data sets: Dataport (Pecan Street Inc, 2019a), REDD (Kolter & Johnson, 2011), UK-DALE (Kelly & Knottenbelt, 2015b), iAWE (N. Batra, Gulati, Singh, & Srivastava, 2013), AMPds (Makonin, Popowich, Bartram, Gill, & Bajić, 2013), and Smart\* Barker et al. (n.d.) . The NILMTK converters allow relevant metadata and datasets to be stored in a Hierarchical Data Format (HDF5), which enables users to organize data structured in a binary format. Not only the electricity data, but NILMTK data format also can store related metadata such as location and maximum sample period, to help enhance the prediction of NILM. The new version of the converter for Dataport is required as the currently provided converter is not applicable for recent datasets (P. Batra Kelly, 2019a). The implementation of the new converter will be introduced in the next Chapter.

### 3.5.2 NILM Algorithms

Currently, there are four NILM algorithms available in NILMTK platform: Combinatorial Optimization (CO) (Korte & Vygen, 2012), Factorial Hidden Markov Model (FMHH) (Ghahramani & Jordan, 1995), Hart's 1985 Algorithm (Hart85) (Hart, 1992b), and Maximum Likelihood Estimation (MLE) (Hart, 1992a). In this study, Combinatorial optimization (CO) and Factorial hidden Markov model (FHMM) algorithms are adopted to run on the edge device, Raspberry Pi 4 model B. The two algorithms are selected because they are the most commonly used algorithms when it comes to NILM performance comparison when utilizing NILMTK framework. It is known that CO and FMHH show the best performance for the simplest algorithms in speed and error rate (Nguyen, 2020).

#### 3.5.2.1 Combinatorial optimization (CO)

CO looks for the optimal combination of various appliance states that minimize the gap between the sum of the estimated power of appliance and the observed gross power. CO can be represented by the following state assignment:

$$\hat{x}_t^{(n)} = \underset{\hat{x}_t^{(n)}}{\operatorname{argmin}} \left| \bar{y} - \sum_{i=1}^n \hat{x}_t^{(n)} \right|$$

(N. Batra et al., 2014) In CO, each time slice is considered to be independent because each time slice is assumed as a single optimization problem. Also, CO is NP-complete as it is similar to the subset sum problem. The time complexity of energy disaggregation is  $O(TS^N)$ , where  $T$  is given time slices,  $S$  is the number of states of appliances, and  $N$  is the number of electric devices.

3.5.2.2 Factorial hidden Markov model (FHMM) An FHMM model is an extended version of Hidden Markov Model (Ghahramani & Jordan, 1995). A couple of HMMs are evolved independently in parallel, which turns out to joint all the hidden states. The complexity of energy disaggregation for FHMM is  $O(TS^{2N})$ , as a result of taking three parameters: prior probability, transition matrix, and emission matrix.



### 3.5.3 EV profile extraction algorithm

In order to secure the concise EV charging profiles, the noises will be filtered out by adapting the sliding-window technique (see 3.1. *pred\_CO* and *pred\_FHMM* refer to predictions from CO classier and FHMM classifier respectively. The size of the window ('100s'), a maximum mean value of the window ('5000'), and minimum duration of EV charging profile ('20minutes') can be adjusted depending on the traits of the monitored electric vehicle. Finally, the start, end, and elapsed times of EV charging load will be recorded and exported to the cloud along with the generated plots from the NILM classifiers.

---

**Algorithm 3.1** EV charging Load Extraction

---

```
1: function EXTRACT_EV( pred_CO, pred_FHMM)
2:   // sotre EV predictions to the new data frames
3:   df_CO; pred_CO ['Electric vehicle']
4:   df_FHMM; pred_FHMM ['Electric vehicle']
5:   // sliding window=100s
6:   df_CO=df_CO[df_CO.rolling('100s').mean() > 5000]
7:   df_FHMM=df_FHMM[df_FHMM.rolling('100s').mean() > 5000]
8:
9:   times_CO = {}; times_FHMM = {}
10:  i=0; j=0;
11:  start = first timestamp of df_CO
12:  //pair up of start and end time of a single EV charging load
13:  while i < not len(df_CO)-1 do
14:    if df_CO[i+1] - df_CO[i] > 20minutes then
      times_CO[start] = i_th timestamp; start = df_CO[i+1]; i+=1 else i+=1
15:    end while
16:    times_CO[start]=df_CO[-1]
17:
18:    //Do the same process for times_FHMM
19:
20:  return times_CO, times_FHMM
21: end function
22: for start, end in times_CO.items() do print(start, end, end-start)
23: end for
24: for start, end in times_FHMM.items() do print(start, end, end-start)
25: end for
```

---

### 3.6 Performance Evaluation Criteria

This research evaluates the effectiveness of the CO and FHMM classifiers for NILM by measuring Root Mean Square Error (RMSE). Additionally, the extraction algorithm for EV profiles can be assessed by four values of Confusion Matrix: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

#### 3.6.1 Root Mean Square Error (RMSE)


Root Mean Square Error (RMSE) is the standard deviation of the prediction errors (see the equation below). This can be calculated by measuring the distance between the regression line data points and the prediction results. RMSE are generally employed in model evaluation researches (Chai & Draxler, 2014).

$$rmse = \sqrt{\left(\frac{1}{n}\right) \sum_{i=1}^n (y_i - x_i)^2}$$

#### 3.6.2 Runtime

Along with accuracy, this study considers the performance of the algorithms by time spent until getting the results. The elapsed time to train classifiers and disaggregate energy loads with different amount of datasets will be measured in every test. This information will help optimize the test conditions for the best performance to extract EV charging loads.

Table 3.3. *System Requirements of OwnCloud*

OwnCloud	Technical Specs
	<ul style="list-style-type: none"> <li>• OS: Debian 7, 8, and 9</li> <li>• Database: MySQL or MariaDB 5.5+</li> <li>• Web server: Apache 2.4 with prefork and mod_php</li> <li>• PHP Runtime: 5.6, 7.0, 7.1, and 7.2</li> </ul>

### 3.7 Adoption of Cloud Server Platform

After the edge computing phase, the results are sent out to the cloud server platform. OwnCloud is a widely used cloud platform as a free file-hosting application. This platform is adopted to store and manage the output data generated by the NILM algorithms of Raspberry Pi. The system requirements to use OwnCloud on Raspberry Pi are displayed in Table 3.3.

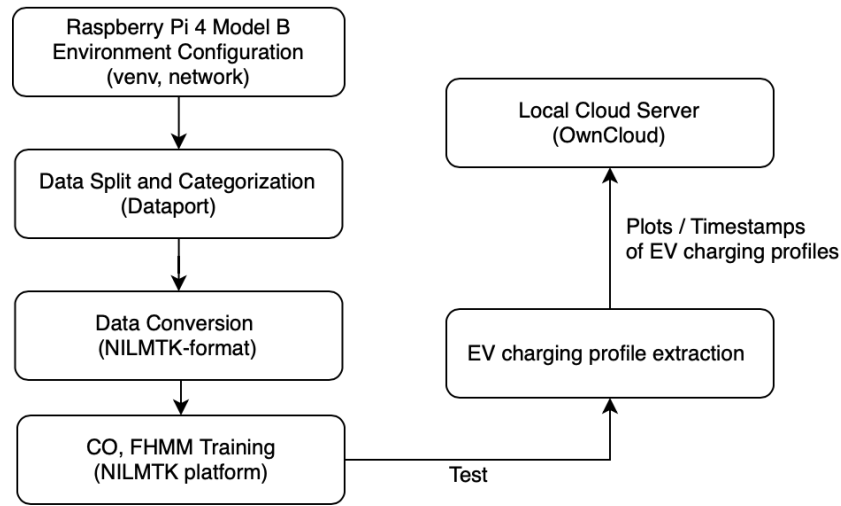
### 3.8 The flowchart of Experiment Scenario

Figure 3.4 displays a flowchart for this research. The data fed to the edge device are 1-second, 1-minute, and 15-minute based Dataport. The data with high sampling frequency (1-second based data) are split into several pieces to allow Raspberry Pi to load and categorize the data with assigned house IDs.

Next, the data will be preprocessed to be imported to the NILM toolkit (NILMTK) platform. Depending on whether solar-generated power is considered or not, the total load has to be stored in HDF5 file differently. If it is not considered, the sum of power consumption of electric appliances becomes the total load. If it is considered, the power measured on the grid subtracted by solar-generated power becomes the total load.

After converting datasets into NILMTK-format, two classifiers-Combinatorial Optimization (CO) and Factorial Hidden Markov Model (FHMM) trained and tested data as NILM algorithms. The experiments were conducted under different conditions: the sampling frequency of data, the number of data points, consideration of solar influx.

Finally, the outputs of the experiment were displayed, stored, and exported to a local cloud server. The outputs include Root Mean Squared Error (RMSE) as one of accuracy metrics, plots for ground-truths and predictions of EV charging loads, and a list of information that contains start/end/elapsed time of EV charging loads based on the predictions.



*Figure 3.4. An end-to-end system flow*

### 3.9 Summary

This chapter described the system design as well as open public datasets for this study. In addition, NILM algorithms and evaluation methods for algorithm performance were investigated.

## CHAPTER 4. EXPERIMENT

This chapter describes tools and specific mechanisms while conducting the experiment based on what was introduced in the previous chapters. The details of the experiment environment and result for this research will be analyzed and discussed in this chapter.

### 4.1 Dataset and Features

As described in Section 3.3, this research uses Dataport dataset from Pecan Street Inc. The changed access policy of Pecan Street Inc. allows six months of data, captured every 1-second, 1-minute, and 15-minute energy datasets from three regions, which are New York, California, and Austin, to a university member.

The individual home is identified with its unique id number, and each house has 75 columns for electric devices, which are recorded to a smart meter. Datasets collected from 25 homes in each region were accessible for this experiment.

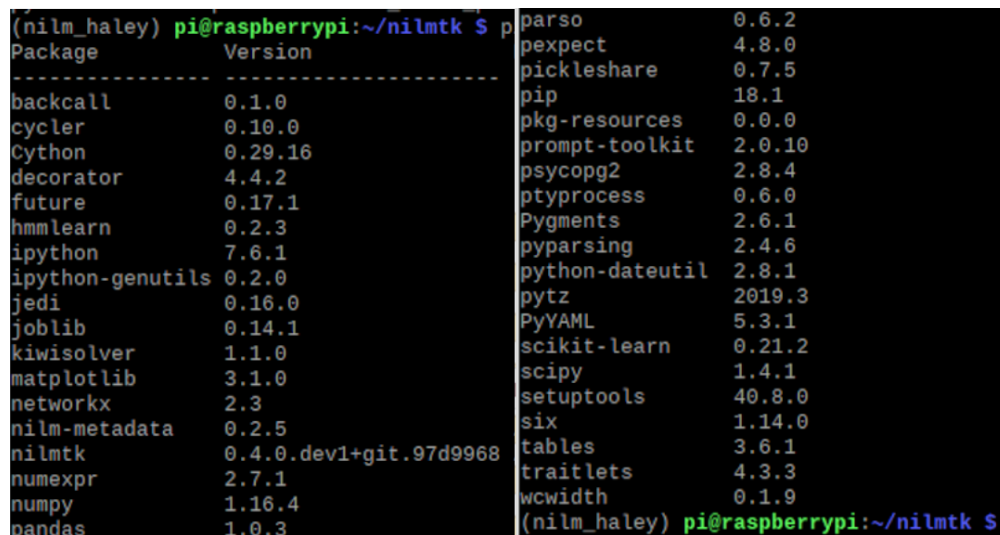
### 4.2 Experimental Environment

The experiment was conducted upon Raspberry Pi 4 Model B, which is one of the most recent models from Raspberry Pi Foundation. This section covers the required libraries and cloud server setup for Raspberry Pi.

#### 4.2.1 Libraries used on Raspberry Pi

The versions of Non-Intrusive Load Monitoring Toolkit (NILMTK) and its metadata are 0.4.0 and 0.2.5 respectively. NILMTK requires at least Python 3.6, and Python 3.7 is used on Raspberry Pi. When a prototype experiment was conducted on MacOS, a virtual environment of Anaconda was used, which bundles most of the NILMTK's required packages together.

On the other hand, Miniconda, which is a minimized installer for conda, had to be used on Raspberry Pi due to its system environment. The problem was that conda installed on Raspberry Pi does not support full packages of what conda-forge provides in order to install NILMTK. After a couple of trials and errors, a virtual environment using ‘venv’ made it possible to install NILMTK on Raspberry Pi successfully. While NILMTK was installed, 15 packages (cython, future, hmmlearn, ipython, matplotlib, networkx, numpy, pandas, prompt-toolkit, psycpgz, scikit-learn, scipy, setuptools, six, tables) were manually installed using pip3 and each package version was matched to that of MacOS environment. All packages installed in the virtual environment are shown in Figure 4.3.



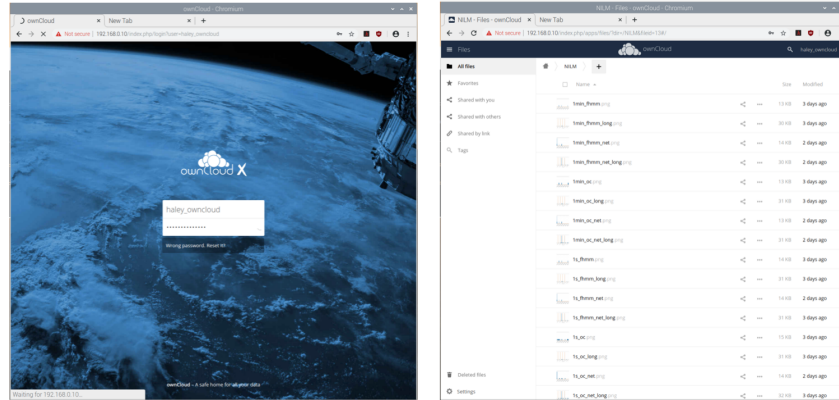
The image shows a terminal window with a dark background. The prompt is `(nilm_haley) pi@raspberrypi:~/nilmtk $`. The user has run a command to list installed packages and their versions. The output is displayed in two columns. The first column lists the package names, and the second column lists the version numbers. The packages are: backcall, cython, decorator, future, hmmlearn, ipython, ipython-genutils, jedi, joblib, kiwisolver, matplotlib, networkx, nilm-metadata, nilmtk, numexpr, numpy, pandas, parso, pexpect, pickleshare, pip, pkg-resources, prompt-toolkit, psycpg2, ptyprocess, Pygments, pyparsing, python-dateutil, pytz, PyYAML, scikit-learn, scipy, setuptools, six, tables, traitlets, and wcwidth.

Package	Version
backcall	0.1.0
cython	0.10.0
Cython	0.29.16
decorator	4.4.2
future	0.17.1
hmmlearn	0.2.3
ipython	7.6.1
ipython-genutils	0.2.0
jedi	0.16.0
joblib	0.14.1
kiwisolver	1.1.0
matplotlib	3.1.0
networkx	2.3
nilm-metadata	0.2.5
nilmtk	0.4.0.dev1+git.97d9968
numexpr	2.7.1
numpy	1.16.4
pandas	1.0.3
parso	0.6.2
pexpect	4.8.0
pickleshare	0.7.5
pip	18.1
pkg-resources	0.0.0
prompt-toolkit	2.0.10
psycpg2	2.8.4
ptyprocess	0.6.0
Pygments	2.6.1
pyparsing	2.4.6
python-dateutil	2.8.1
pytz	2019.3
PyYAML	5.3.1
scikit-learn	0.21.2
scipy	1.4.1
setuptools	40.8.0
six	1.14.0
tables	3.6.1
traitlets	4.3.3
wcwidth	0.1.9

Figure 4.1. Packages used for NILMTK on Raspberry Pi

### 4.2.2 Cloud Server Setup for Raspberry Pi

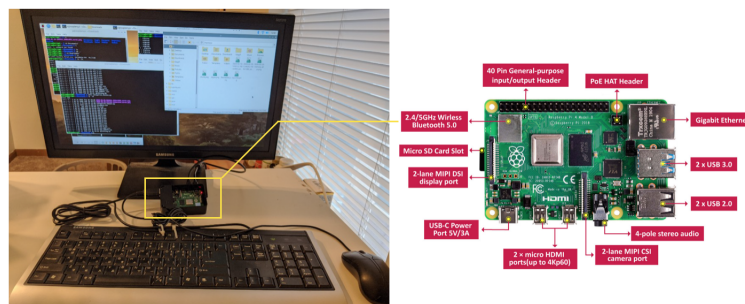
OwnCloud was adopted to sync and manage data of Raspberry Pi. Since OwnCloud runs on LAMP (Linux, Apache, MySQL/MariaDB, PHP) stack (Cannon, 2014), the environment was configured on Raspberry Pi. Figure 4.2 shows the successful setup and data sharing on OwnCloud.



*Figure 4.2.* Data Exporting to the cloud server of Raspberry Pi

### 4.2.3 Location

The location of the experiment does not matter as long as wireless internet access is secured because this experiment uses static time-series datasets already captured by smart meters. Therefore, the experiment was conducted in an indoor environment, as shown in Figure 4.3.



*Figure 4.3.* Experimental Environment and Raspberry Pi 4 B

## 4.3 Experimental Result

This section covers the results of data preprocessing, outputs of NILM algorithms for recognizing and identifying EV charging profiles upon different conditions.

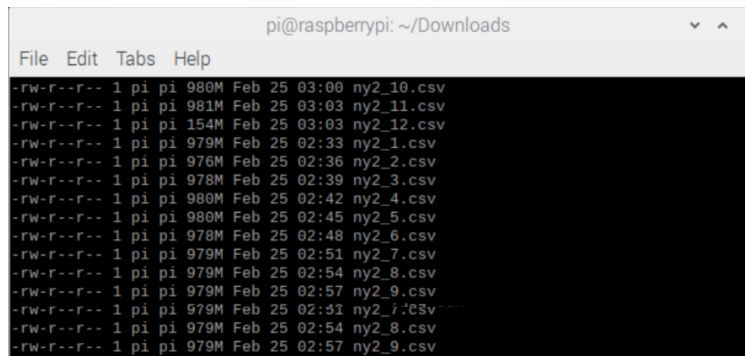
### 4.3.1 Data Preprocessing

Data Preprocessing consists of three steps: Data Splitting for 1-second datasets, Data Categorizing to separate each household data, and finally Data Converting to NILMTK format.

#### Step 1 Data Splitting

Dataport contains time-series datasets, obtained every 1 second, 1 minute, and 15 minutes. As for 1-second datasets, the size of an extracted file exceeds 10GB. Not surprisingly, one of the 1-second dataset collected in New York area ('1-second NY2 file') is 11.4GB, which contains 66,938,399 data points.

On Raspberry Pi, however, memory error commonly occurs when processing a file exceeding 1GB. To address this problem, one large file was split into twelve separate files, which contain 6 million rows (1.1GB) per each, as shown in Figure 4.4.



```
pi@raspberrypi: ~/Downloads
File Edit Tabs Help
-rw-r--r-- 1 pi pi 980M Feb 25 03:00 ny2_10.csv
-rw-r--r-- 1 pi pi 981M Feb 25 03:03 ny2_11.csv
-rw-r--r-- 1 pi pi 154M Feb 25 03:03 ny2_12.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:33 ny2_1.csv
-rw-r--r-- 1 pi pi 976M Feb 25 02:36 ny2_2.csv
-rw-r--r-- 1 pi pi 978M Feb 25 02:39 ny2_3.csv
-rw-r--r-- 1 pi pi 980M Feb 25 02:42 ny2_4.csv
-rw-r--r-- 1 pi pi 980M Feb 25 02:45 ny2_5.csv
-rw-r--r-- 1 pi pi 978M Feb 25 02:48 ny2_6.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:51 ny2_7.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:54 ny2_8.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:57 ny2_9.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:57 ny2_10.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:54 ny2_8.csv
-rw-r--r-- 1 pi pi 979M Feb 25 02:57 ny2_9.csv
```

Figure 4.4. Specifying types of columns to Pandas

The splitting task was done on a macOS laptop since this task needs only for 1-second datasets, not for 1-minute and 15-minute datasets. This was to reduce the time consumed when loading 1-second datasets. From the following part, these twelve files are also assumed as “raw files”.



## Step 2 Data Categorizing

After feeding data into Raspberry Pi, the maximum size of a file is approximately 1GB. Each file was then categorized into files which contain datasets for individual houses. In order to optimize the data, houses that have empty data on electric vehicle charging load column were filtered out. The unique home IDs in the remaining datasets were retrieved, and the '1-second NY2 file' has only five homes (ID: 27, ID: 1222, ID: 3000, ID: 5679, and ID: 9053) containing electric vehicle charging loads.

The challenge here is that 1GB is still too big when loaded on Raspberry Pi at once. To handle this issue, all the column types are specified to reduce loads of Pandas, as shown in Figure 4.6. Nevertheless, the memory error still occurred the device.

```
fname = 'ny2_1.csv'
types = {'dataid':np.dtype(int64), 'air1':np.dtype(float), 'air2':np.dtype(float), 'air3':np.dtype(float), \
'airwindowunit1':np.dtype(float), 'aquarium1':np.dtype(float), 'bathroom1':np.dtype(float), 'bathroom2':np.dtype(float), \
'bedroom1':np.dtype(float), 'bedroom2':np.dtype(float), 'bedroom3':np.dtype(float), 'bedroom4':np.dtype(float), \
'bedroom5':np.dtype(float), 'battery1':np.dtype(float), 'car1':np.dtype(float), 'car2':np.dtype(float), \
'circumpump1':np.dtype(float), 'clotheswasher1':np.dtype(float), 'clotheswasher_dryg1':np.dtype(float), \
'diningroom1':np.dtype(float), 'diningroom2':np.dtype(float), 'dishwasher1':np.dtype(float), 'disposal1':np.dtype(float), \
'drye1':np.dtype(float), 'dryg1':np.dtype(float), 'freezer1':np.dtype(float), 'furnace1':np.dtype(float), \
'furnace2':np.dtype(float), 'garage1':np.dtype(float), 'garage2':np.dtype(float), 'grid':np.dtype(float), \
'heater1':np.dtype(float), 'heater2':np.dtype(float), 'heater3':np.dtype(float), 'housefan1':np.dtype(float), \
'icemaker1':np.dtype(float), 'jacuzzi1':np.dtype(float), 'kitchen1':np.dtype(float), 'kitchen2':np.dtype(float), \
'kitchenapp1':np.dtype(float), 'kitchenapp2':np.dtype(float), 'lights_plugs1':np.dtype(float), 'lights_plugs2':np.dtype(float), \
'lights_plugs3':np.dtype(float), 'lights_plugs4':np.dtype(float), 'lights_plugs5':np.dtype(float), \
'lights_plugs6':np.dtype(float), 'livingroom1':np.dtype(float), 'livingroom2':np.dtype(float), 'microwave1':np.dtype(float), \
'office1':np.dtype(float), 'outsidelights_plugs1':np.dtype(float), 'outsidelights_plugs2':np.dtype(float), \
'oven1':np.dtype(float), 'oven2':np.dtype(float), 'pool1':np.dtype(float), 'pool2':np.dtype(float), 'poollight1':np.dtype(float), \
'poolpump1':np.dtype(float), 'pump1':np.dtype(float), 'range1':np.dtype(float), 'refrigerator1':np.dtype(float), \
'refrigerator2':np.dtype(float), 'security1':np.dtype(float), 'sewerpump1':np.dtype(float), 'shed1':np.dtype(float), \
'solar':np.dtype(float), 'solar2':np.dtype(float), 'sprinkler1':np.dtype(float), 'sumppump1':np.dtype(float), \
'utilityroom1':np.dtype(float), 'venthood1':np.dtype(float), 'waterheater1':np.dtype(float), 'waterheater2':np.dtype(float), \
'wellpump1':np.dtype(float), 'winecooler1':np.dtype(float), 'leg1v':np.dtype(float), 'leg2v':np.dtype(float)}
```

Figure 4.5. Specifying types of columns to Pandas

The second trial was to make chunks containing 100,000 data points when loading a file and this worked, as shown in Figure 4.6.

```
df_chunk = pd.read_csv(fname, chunksize = 100000) #dtype=types)
for i, chunk in enumerate(df_chunk):
    df = chunk[chunk['car1'].notna()]
    df_rows_5679=df.loc[df['dataid']==5679]
    df_rows_27=df.loc[df['dataid']==27]
    if i == 0:
        df_rows_5679.to_csv(r'ny2_id5679_.csv', index=False, header=True)
        df_rows_27.to_csv(r'ny2_id27_.csv', index=False, header=True)
    else:
        df_rows_5679.to_csv(r'ny2_id5679_.csv',mode = 'a' , index=False, header=False)
        df_rows_27.to_csv(r'ny2_id27_.csv',mode = 'a' , index=False, header=False)
```

Figure 4.6. Making chunks of data on Raspberry Pi

### Step 3 Data Converting to NILMTK format

As described in Section 3.5, NILMTK uses its own data format to store power data as well as the metadata based on the Hierarchical Data Format version 5 (HDF5) binary file format. The initial step for using NILMTK is to convert the datasets to the NILMTK HDF5 format. Despite the fact that NILMTK contains dataset converters for commonly used public datasets for NILM, the converter for Dataport is outdated. Pecan Street inc. requires an additional security verification to allow users to access to Dataport by sending six-digit code to the account. This made it hard to have direct access from the edge device using the currently provided converter for Dataport provided by NILMTK. This problem was addressed by retrofitting the existing converter codes and modifying metadata for Dataport. The mechanism of the new converter is divided into three parts: to map data loads to their similar type to categorize a number of appliances, to separate total load metadata and appliance metadata, and to store modified data frames into a h5 file.

---

**Algorithm 4.1** A new NILMTK converter for Dataport.

---

```
1: //Data loads mapping
2: feed_mapping = 'air1': 'type': 'air conditioner', 'air2': 'type': 'air conditioner', 'car1': 'type':
   'electric vehicle', 'kitchen1': 'type': 'sockets', 'livingroom1': 'type': 'sockets', 'room':
   'living room', 'livingroom2': 'type': 'sockets', 'room': 'living room', 'heater1': 'type':
   'electric space heater', 'refrigerator1': 'type': 'fridge', 'refrigerator2': 'type': 'fridge', 'solar':
   'type': 'solar', 'grid': 'type': 'grid', etc
3: function CONVERT_CSV_TO_H5(csv_filename, h5_filename)
4:     //CSV data load
5:     //Data sorted by 'localminute'
6:     //Open h5 file
7:     function DATAFRAME_TO_HDF(dataframes, building_id, h5_store, timestamp_name)
8:         //Set timestamp as index of dataframe
9:         //Create building metadata for yaml file (building id, total loads, appliances load)
10:        //Convert timeseries data into dataframe
11:        //Modify column names
12:        //Store dataframe in hdf5
13:        //Separate total load metadata and appliance metadata
14:        //Write building metadata to a yaml file
15:    end function
16:    //Close h5 file
17:    //Write yaml file to h5 file
18: end function
```

---

#### 4.3.2 Data Analysis Result

ID=5679 has seven electric appliances (Air Conditioner, four sockets (bedroom1, garage1, kitchenapp1, kitchenapp2), electric furnace, and electric vehicle) after dropping the N/A columns. The maximum power load of ID=5679 is an electric vehicle, which hits almost 7kW (see Figure 4.7). The maximum loads of the air conditioner and the remainings are almost 2kW and 1kW.

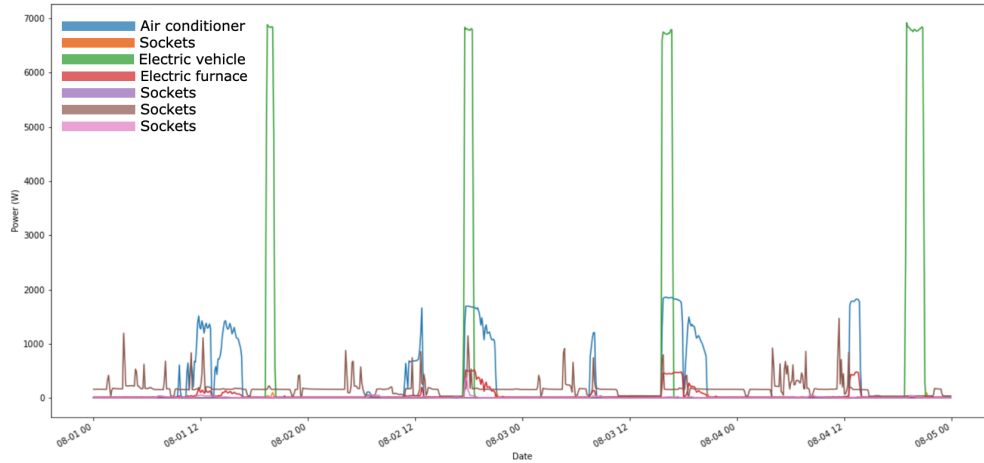


Figure 4.7. Appliances graph from ID=5679

The vast majority of the pie chart (see Figure 4.8) consists of EV (Green), AC (Blue), and one of the sockets. Even though the maximum power value of EV in ID=5679 is 6.992kW, the sum of power usage amount of AC exceeds that of EV due to usage frequencies.

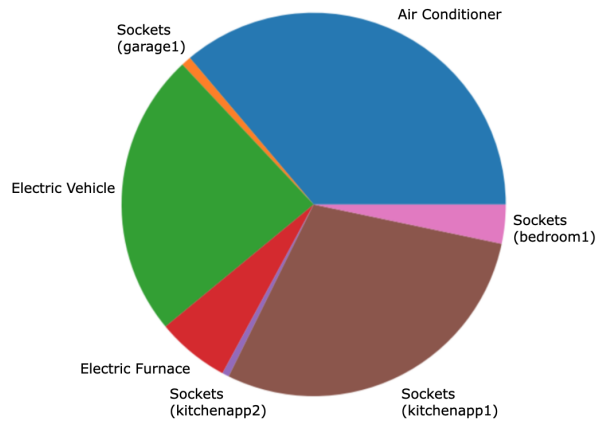
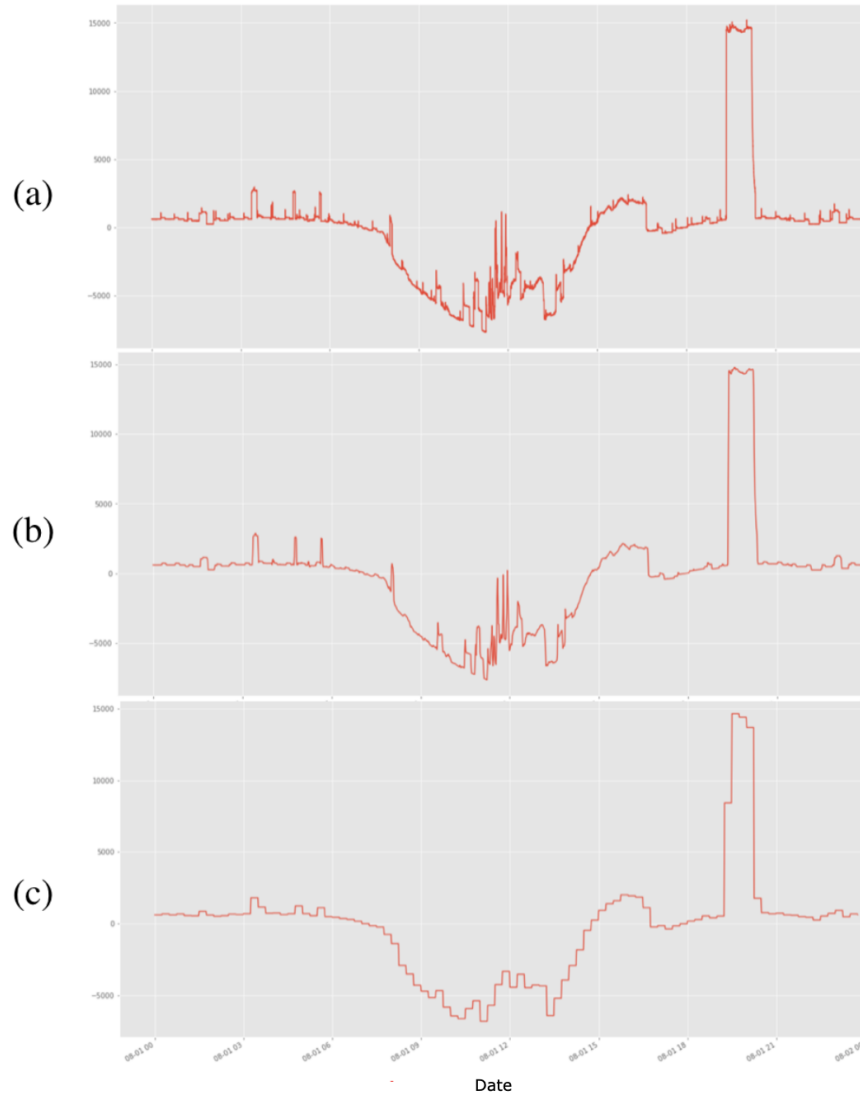


Figure 4.8. A pie chart of power data from ID=5679 in August 2019

The plots of 1-second, 1-minute, and 15-minute datasets obtained between August 1 and 2 are displayed in Figure 4.9. 1-second data have high sampling frequency and much noises at the same time, 1-minute data shows smoothen graph, and 15-minute data seems to have a loss of features.



*Figure 4.9.* (a) 1-second, (b) 1-minute, (c) 15-minute power data from ID=5679

Figure 4.10 shows 1-minute power data plots for ID:5679 datasets collected between August 1 and 5, 2019. The ‘Site meter’ (Grey) in Figure 4.10(a) indicates the sum of the power consumption of the individual appliances ((*sum\_power*), including EV loads while ‘Site meter’ (Grey) in Figure 4.10(b) represents the total power consumption subtracted by solar-generated power ((*net\_power*)).

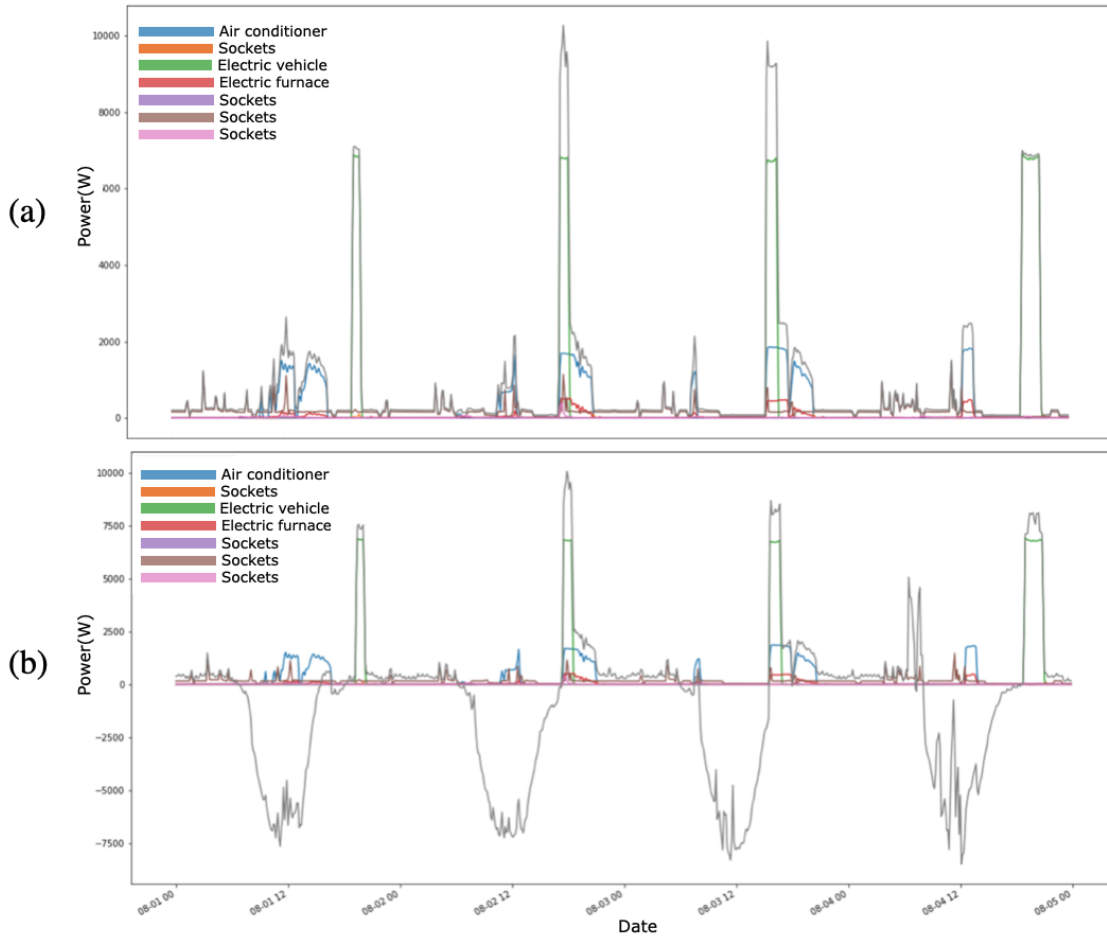


Figure 4.10. 1-minute Power data from ID=5679. (a) ‘Site meter’= the sum of power consumption of electric appliances (*sum\_power*), (b) ‘Site meter’= power measured on the grid - solar generated power (*net\_power*)

### 4.3.3 NILM result for EV charging loads

The 1-second, 1-minute, and 15-minute datasets of Dataport were trained and tested using Combinatorial Optimization (CO) and Factorial Hidden Markov Model (FHMM) algorithms on the Raspberry Pi 4 Model B. The experiments were conducted based on short-term data (10 days) and long-term data (30 days) with approximately 80% and 20% of training and test data. The outputs of the experiments are Root Mean Squared Error (RMSE) as one of accuracy metrics, two images that display the ground truth and predictions of CO, FHMM for EV charging loads, and list of information that contains start/end/elapsed time of EV charging loads based on predictions.

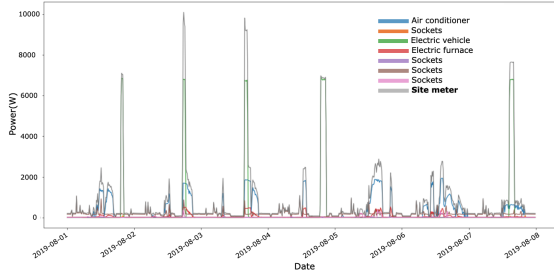
#### 4.3.3.1 NILM using short-term data

The short-term data were obtained from ID=5679 between August 1 and 10, 2019: Data between August 1 and 8 are used for the training set, data between August 8 and 10 are used for test data.

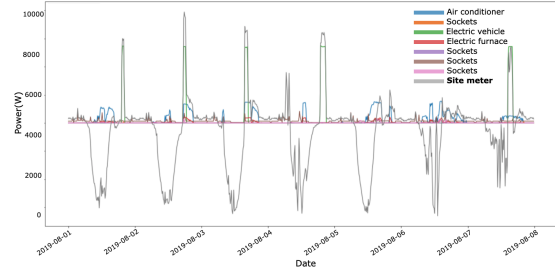
The first experiment using the short-term data is based on the sum of power consumption of electric appliances, including EV loads (*sum\_power*). The ‘Site meter’ (Grey) as shown in Figure 4.11(a) indicates that the accumulated appliances power loads (*sum\_power*), as well as each appliance power profile, are trained to identify EV charging loads in the *sum\_power* test data set (see Figure 4.11(c)). The second experiment using short-term data is based on the total subtracted by solar-generated power (*net\_power*). The ‘Site meter’ (Grey) as shown in Figure 4.11(b) indicates that the *net\_power* and each appliance power profile are trained to identify EV charging loads in the *net\_power* test data set (see Figure 4.11(d)). The ground truths of EV charging loads for both experiments are plotted in Figure 4.11(e). The actual start, end, and elapsed time of the EV charging loads are described in Figure 4.1:

Table 4.1. *The time profiles of EV charging load (short-term data)*

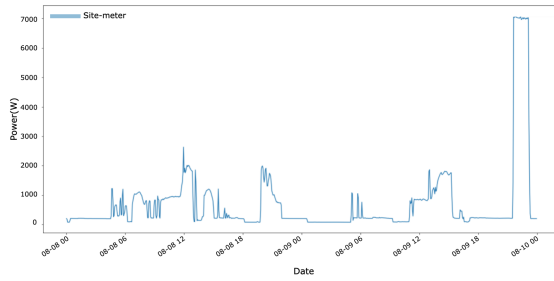
	Start	End	Elapsed Time
1	2019-08-09 21:36:00	2019-08-09 23:11:00	01:34:30



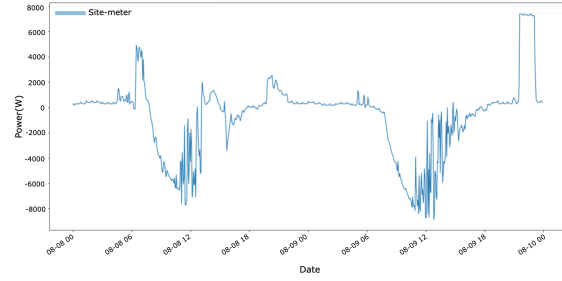
(a) Training data including *sum\_power*



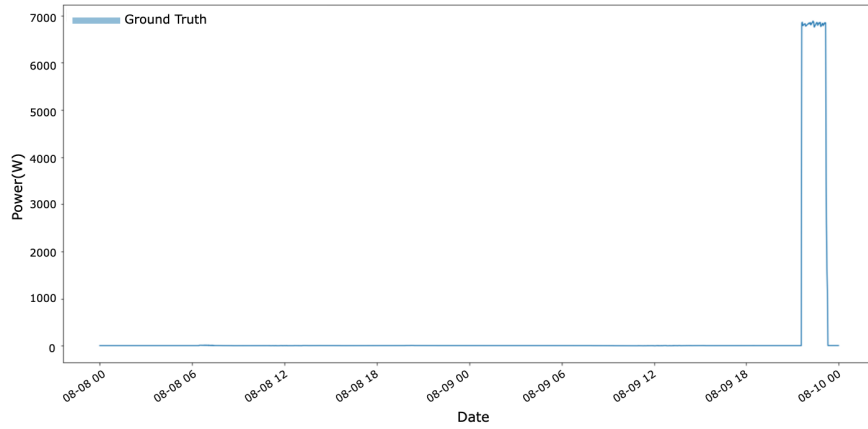
(b) Training data including *net\_power*



(c) Test data *sum\_power*



(d) Test data *net\_power*



(e) The ground truth

Figure 4.11. The short-term data experiments

The results of the short-term data experiments conducted on Raspberry Pi 4 Model B are plotted in Figure 4.12 and 4.13 (The orange plot represents the ground truth while the blue plot represents the predictions). Despite the small amount of training data, most of the test recognized the EV charging load regardless of the number of data points and types of classifiers. As for the experiments using *sum\_power*, all the test showed the correct profile of the single EV charging load (see Figure 4.12(a), 4.12(c), and 4.12(e) for CO, Figure 4.13(a), 4.13(c), and 4.13(e) for FHMM). It is assumed that this is due to the training and test data are clear compared to those of *net\_power*. The tests with 1-second and 1-minute data points showed incorrect recognition (see Figure 4.12(b) and 4.12(d) for CO, Figure 4.13(b) and 4.13(d) for FHMM). In general, the data with higher sampling frequency showed more noises compared to the data with lower sampling frequency.



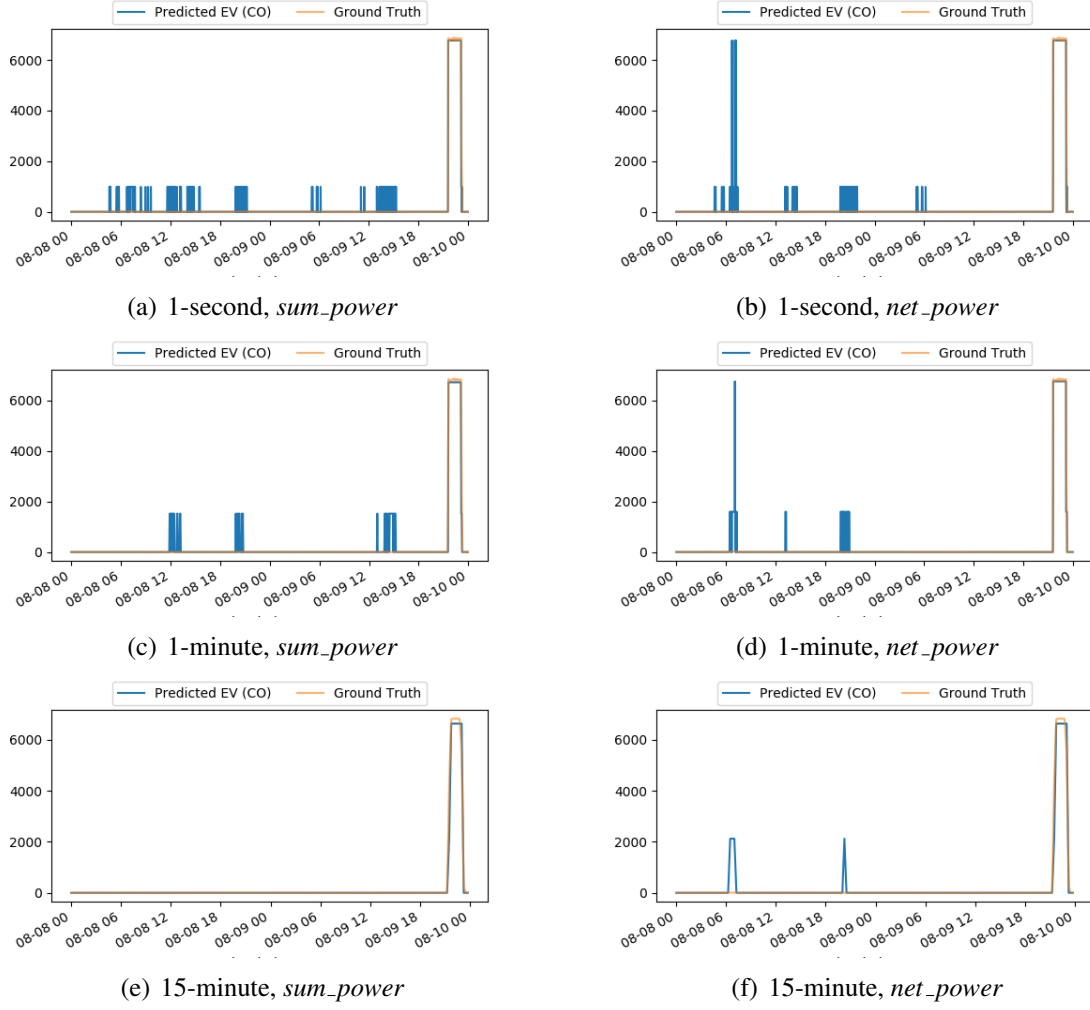


Figure 4.12. The ground truth and predictions of EV charging loads using CO

The Root Mean Square Error (RMSE) for each experiment is described in Table 4.2. The test using 15-minute interval data with the CO classifier showed the best performance for *sum\_power* experiment, while the test using 1-minute interval data with the CO classifier showed the best performance for *net\_power* experiment according to the RMSE.

Table 4.2. RMSE for the short-term data experiment

RMSE	Classifier	1-second		1-minute		15-minute	
		<i>sum_power</i>	<i>net_power</i>	<i>sum_power</i>	<i>net_power</i>	<i>sum_power</i>	<i>net_power</i>
EV	CO	347	400	342	331	181	354
	FHMM	258	381	193	353	205	428

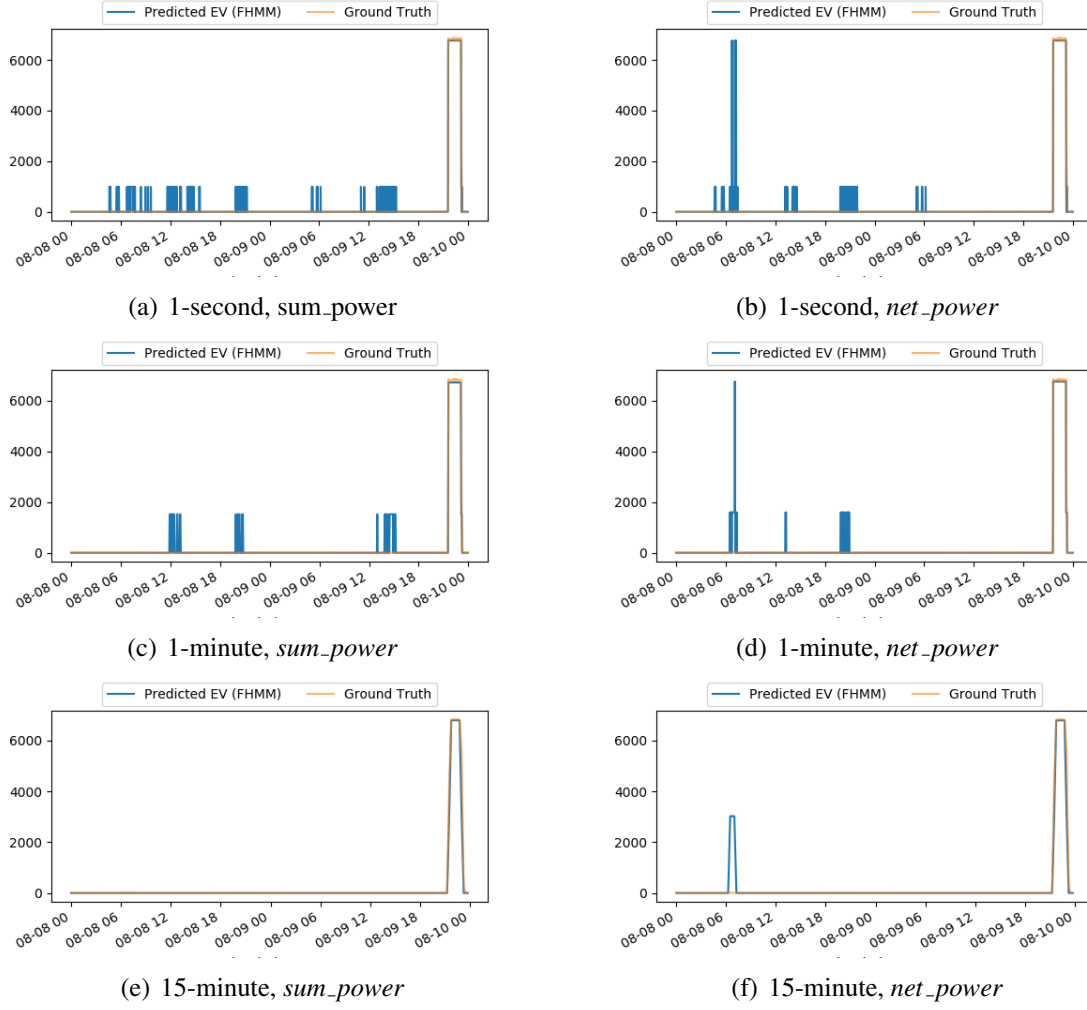


Figure 4.13. The ground truth and predictions of EV charging loads using FHMM

#### 4.3.3.2 NILM using long-term data

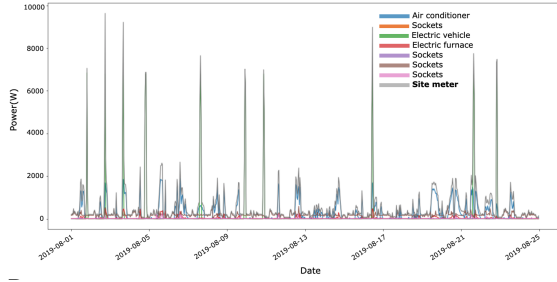
The long-term data were obtained from ID=5679 between August 1 and 30, 2019: Data between August 1 and 25 are used for the training set, data between August 25 and 30 are used for test data.

With the same method of the short-term data experiments, the first experiment using the long-term data is based on the sum of power consumption of electric appliances, including EV loads (*sum\_power*). The ‘Site meter’ (Grey), as shown in Figure 4.14(a) indicates that the accumulated appliances power loads (*sum\_power*), as well as each appliance power profile, are trained to identify EV charging loads in the *sum\_power* test data set (see Figure 4.14(c)). The second experiment using the long-term data is based on the power measured on the grid subtracted by solar-generated power (*net\_power*). The ‘Site meter’ (Grey), as shown in Figure 4.14(b) indicates that the *net\_power* and each appliance power profile are trained to identify EV charging loads in the *net\_power* test data set (see Figure 4.14(d)).

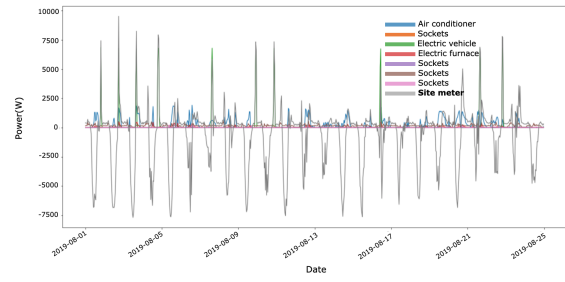
The ground truths of EV charging loads for both experiments are plotted in Figure 4.14(e). The actual start, end, and elapsed time of the four EV charging load during the long-term are described in Table 4.3. This information that contains the time frame of EV charging profiles will be used to discuss the final outputs of each test.

Table 4.3. *The time profiles of EV charging load (long-term data)*

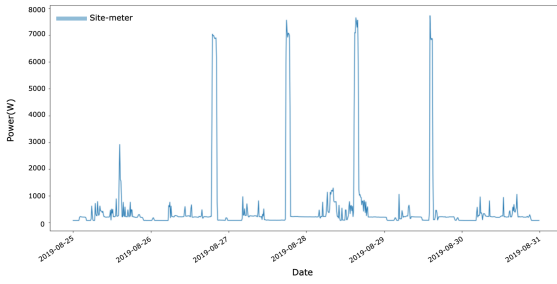
	Start	End	Elapsed Time
1	2019-08-26 18:57:00	2019-08-26 20:25:00	01:28:00
2	2019-08-27 17:45:00	2019-08-27 19:02:00	01:17:00
3	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00
4	2019-08-29 14:10:00	2019-08-29 15:04:00	00:54:00



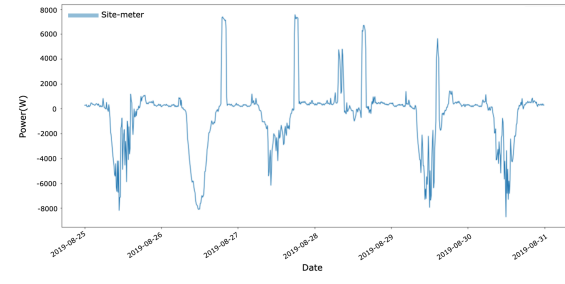
(a) Training data including *sum\_power*



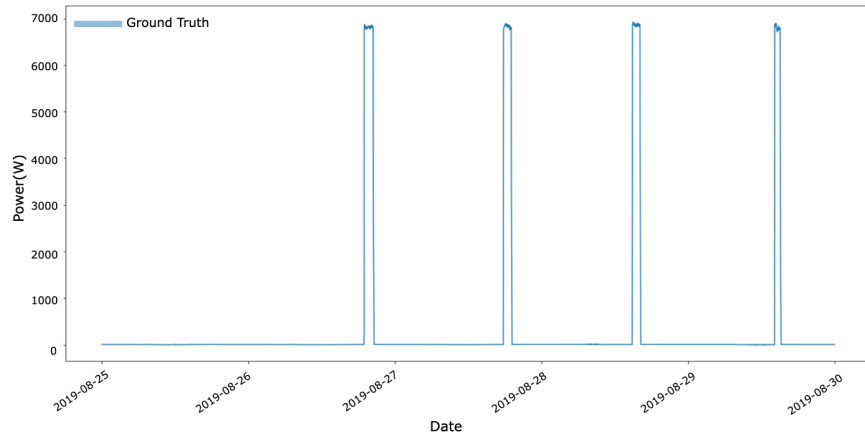
(b) Training data including *net\_power*



(c) Test data *sum\_power*



(d) Test data *net\_power*



(e) The ground truth

Figure 4.14. The long-term data experiments

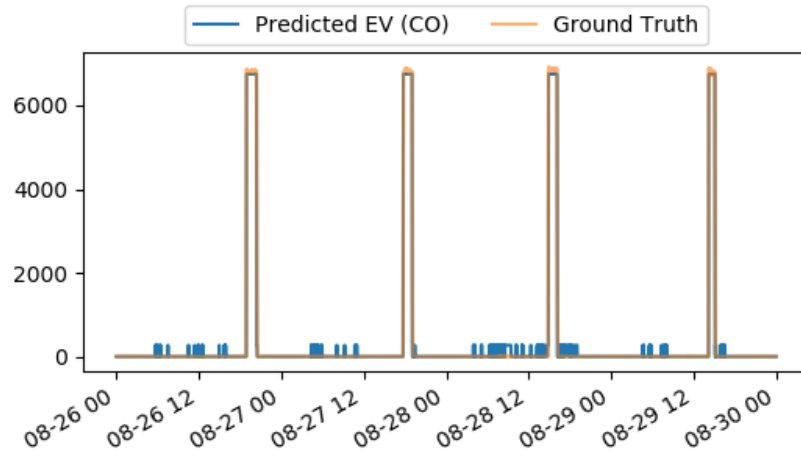
The results of the long-term data experiments conducted on Raspberry Pi 4 Model B using *sum\_power* data are plotted in Figure 4.15 and Figure 4.16 (The orange plot represents the ground truth while the blue plot represents the predictions). All the tests recognized the four EV charging loads regardless of the number of data points and types of classifiers with little noise. In order to secure the concise EV charging profiles, the noises which include consecutive low-power values and high-power weak values are filtered out by adapting sliding-window techniques.

Finally, the time stamps for start, end, and elapsed time of EV charging loads of each test are recorded on Raspberry Pi, as shown in Table 4.5. As introduced in 3.1, whether the data frame is categorized as EV charging profiles or not depends on three parameters: the size of the sliding window, the maximum mean value of the window, and the minimum duration of EV charging load. Currently, the parameters are set as 100 seconds, 5000 watt, and 20 minutes. All the tests show almost identical time frame data despite the different sampling frequency of data.

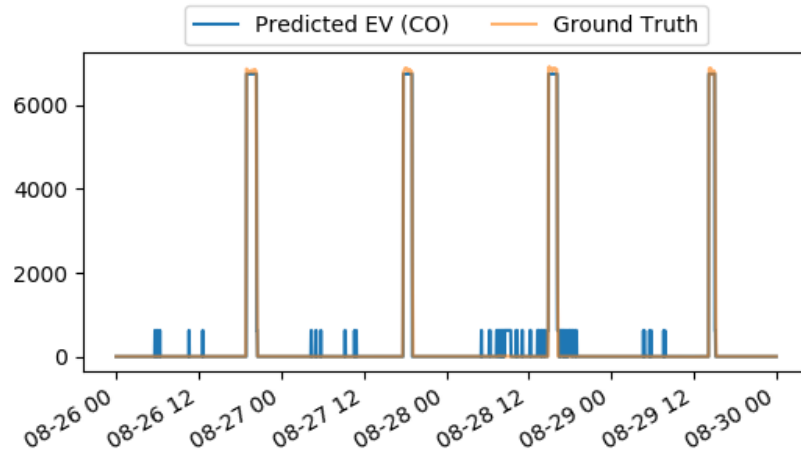
Though all the test recognized four EV charging profiles well, both CO and FHMM classifiers show that the high sampling frequency data performed better than low sampling frequency data (see Table 4.4). Among all, the test using 1-second interval data with the FHMM classifier showed the best performance for *sum\_power* experiment. This result is somewhat different compared to the previous experiment using short-term data as it is shown that the test with low sampling frequency data outperformed that of high sampling frequency data.

Table 4.4. *RMSE for the long-term data experiment based on sum\_power*

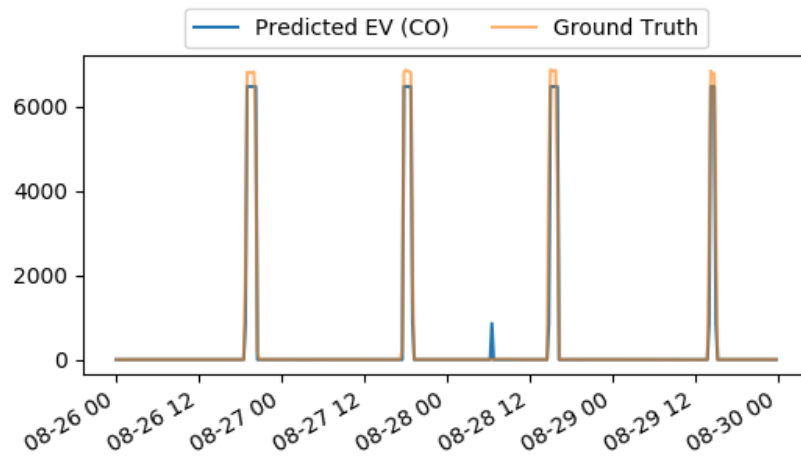
RMSE	Classifier	1-second	1-minute	15-minute
EV	CO	148	160	236
	FHMM	139	160	187



(a) 1-second, *sum\_power*

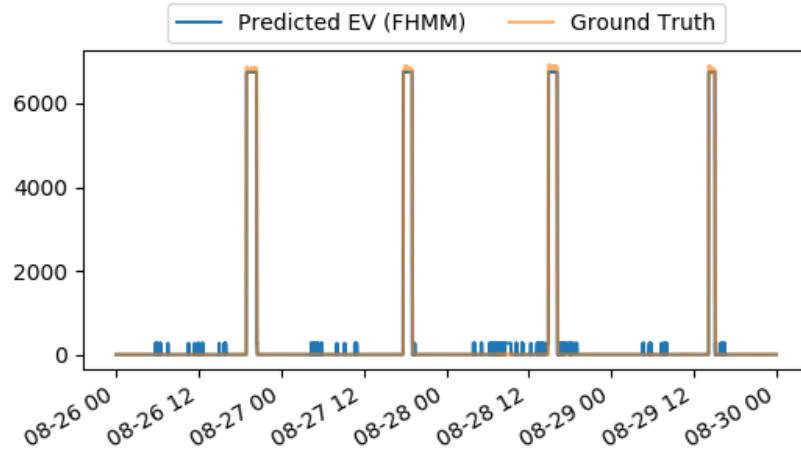


(b) 1-minute, *sum\_power*

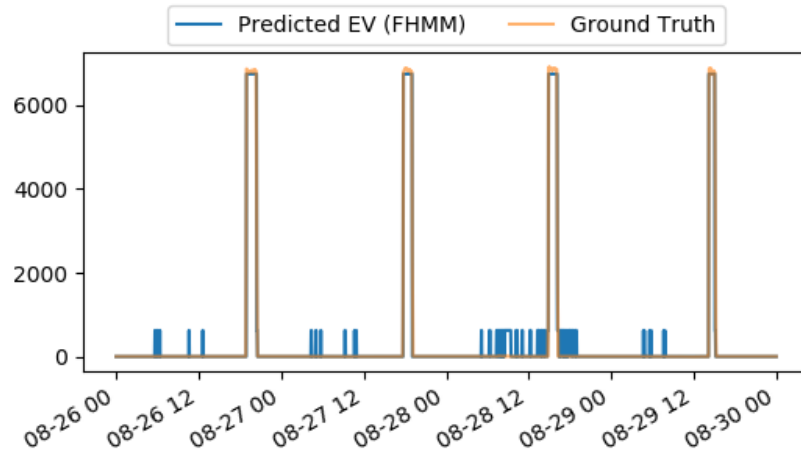


(c) 15-minute, *sum\_power*

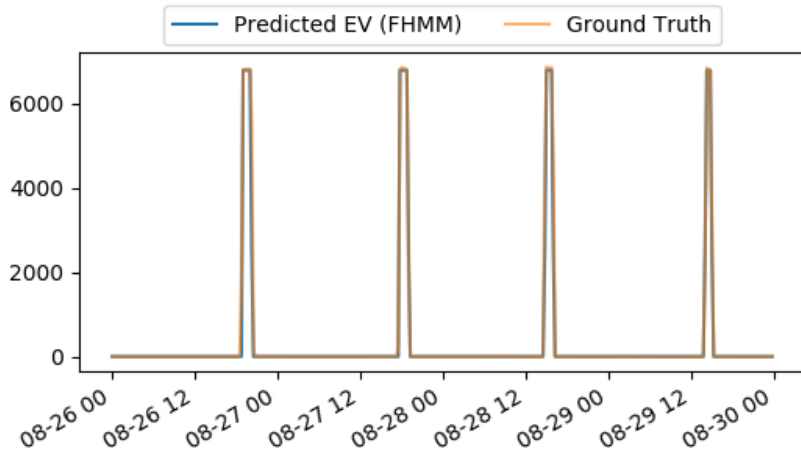
Figure 4.15. The ground truth and predictions of EV charging loads using CO



(a) 1-second, *sum\_power*



(b) 1-minute, *sum\_power*



(c) 15-minute, *sum\_power*

Figure 4.16. The ground truth and predictions of EV charging loads using FHMM

Table 4.5. *The output of EV charging profiles based on sum\_power*

			Start	End	Elapsed Time	T/F
1-sec *	GT	1	2019-08-26 18:57:00	2019-08-26 20:25:00	01:28:00	
		2	2019-08-27 17:45:00	2019-08-27 19:02:00	01:17:00	
		3	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00	
		4	2019-08-29 14:10:00	2019-08-29 15:04:00	00:54:00	
	CO	1	2019-08-26 18:57:15	2019-08-26 20:25:30	01:28:15	T
		2	2019-08-27 17:45:45	2019-08-27 19:02:00	01:16:15	T
		3	2019-08-28 14:52:00	2019-08-28 16:08:15	01:16:15	T
		4	2019-08-29 14:10:30	2019-08-29 15:04:15	00:53:45	T
	FHMM	1	2019-08-26 18:57:15	2019-08-26 20:25:30	01:28:15	T
		2	2019-08-27 17:45:45	2019-08-27 19:02:00	01:16:15	T
		3	2019-08-28 14:51:45	2019-08-28 16:08:15	01:16:30	T
		4	2019-08-29 14:10:30	2019-08-29 15:04:15	00:53:45	T
1-min	CO	1	2019-08-26 18:57:00	2019-08-26 20:24:00	01:27:00	T
		2	2019-08-27 17:46:00	2019-08-27 19:01:00	01:15:00	T
		3	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00	T
		4	2019-08-29 14:10:00	2019-08-29 15:03:00	00:53:00	T
	FHMM	1	2019-08-26 18:57:00	2019-08-26 20:24:00	01:27:00	T
		2	2019-08-27 17:46:00	2019-08-27 19:01:00	01:15:00	T
		3	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00	T
		4	2019-08-29 14:10:00	2019-08-29 15:03:00	00:53:00	T
15-min	CO	1	2019-08-26 19:00:00	2019-08-26 20:00:00	01:15:00	T
		2	2019-08-27 17:45:00	2019-08-27 18:45:00	01:00:00	T
		3	2019-08-28 15:00:00	2019-08-28 15:45:00	01:00:00	T
		4	2019-08-29 14:15:00	2019-08-29 14:45:00	00:30:00	T
	FHMM	1	2019-08-26 19:00:00	2019-08-26 20:00:00	01:00:00	T
		2	2019-08-27 17:45:00	2019-08-27 18:45:00	01:00:00	T
		3	2019-08-28 15:00:00	2019-08-28 15:45:00	00:45:00	T
		4	2019-08-29 14:15:00	2019-08-29 14:45:00	00:30:00	T

\*The sampling period of 1-second data is 15 due to the lack of computing power on Raspberry Pi.



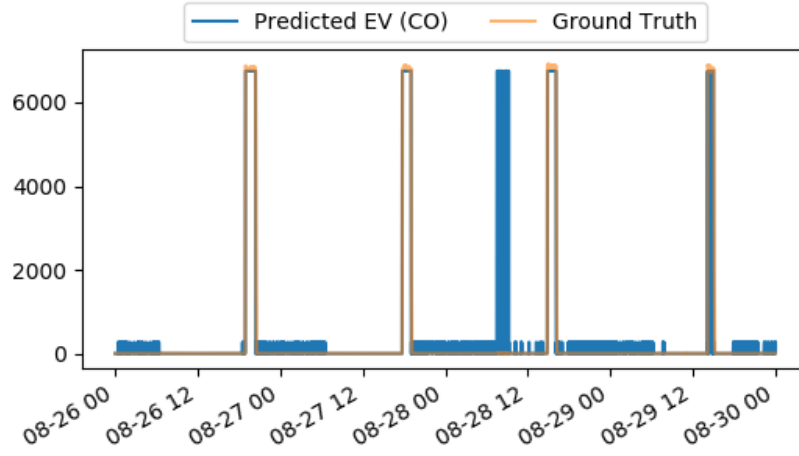
The results of the experiments conducted using *net\_power*, however, showed different aspects than those of *sum\_power*, depending on data points and classifiers as shown in Figure 4.17 and Figure 4.18. Figure 4.17(a) and 4.17(b) from the CO classifier, Figure 4.18(a) and 4.18(c) from the FHMM classifier show a strong signal of non-EV charging power predictions. Figure 4.17(c) shows the non-EV charging power prediction.

The timestamps for start, end, and elapsed time of EV charging loads of each test recorded on Raspberry Pi are shown in Table 4.7. The strong signal of non-EV charging power predictions generated timestamps at around 2019-08-28 07:32:00 and 2019-08-28 08:38:00 with 35 and 37 minutes of elapsed time in 1-second data. The similar aspects are shown on the 1-minute test with 36 and 14 minutes of elapsed time. The non-EV charging power prediction is also identified as EV charging load. The elapsed time is 00:00:00, due to the low sampling frequency of the data points. The 15-minute test using FHMM classifier did not recognize the fourth charging load, and the load was ignored in the extraction algorithm.

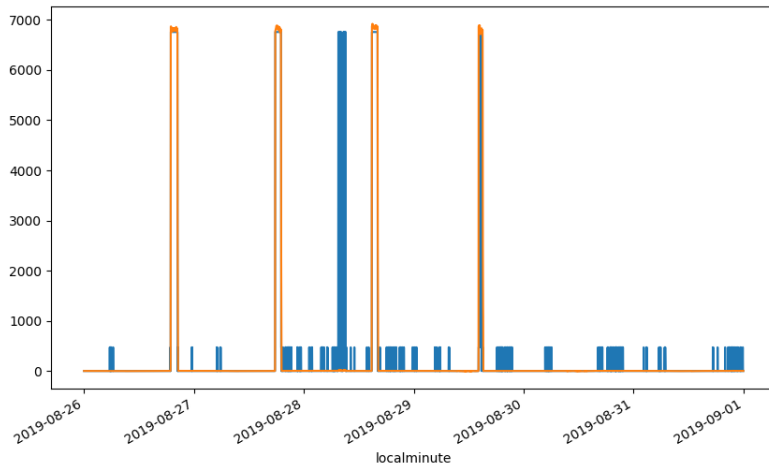
According to Table 4.6, both CO and FHMM classifiers show that the low sampling frequency data performed better than high sampling frequency data. Among all, the test using 15-minute interval data with the FHMM classifier showed the best performance for *net\_power* experiment. This result is similar to experiments using short-term data, showing that the test with low sampling frequency data outperformed that of high sampling frequency data. On the contrary, this result is different compared to the previous *sum\_power* experiment, as it is shown that the test with high sampling frequency data outperformed that of low sampling frequency data.

Table 4.6. *RMSE for the long-term data experiment based on net\_power*

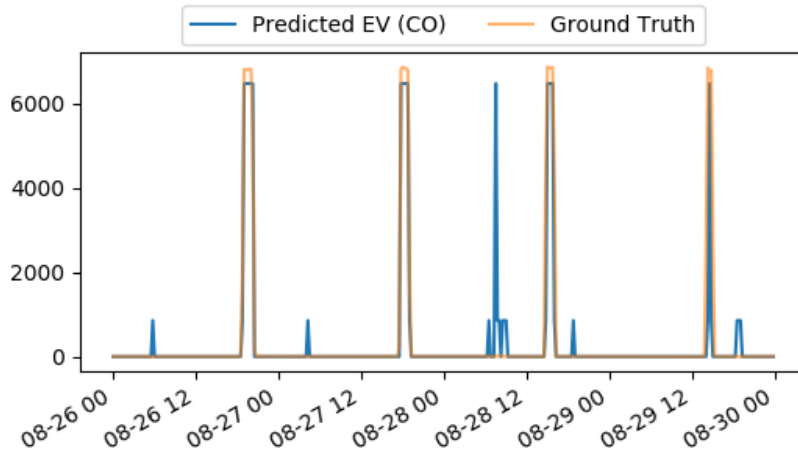
RMSE	Classifier	1-second	1-minute	15-minute
EV	CO	664	600	538
	FHMM	662	598	466



(a) 1-second, *net\_power*

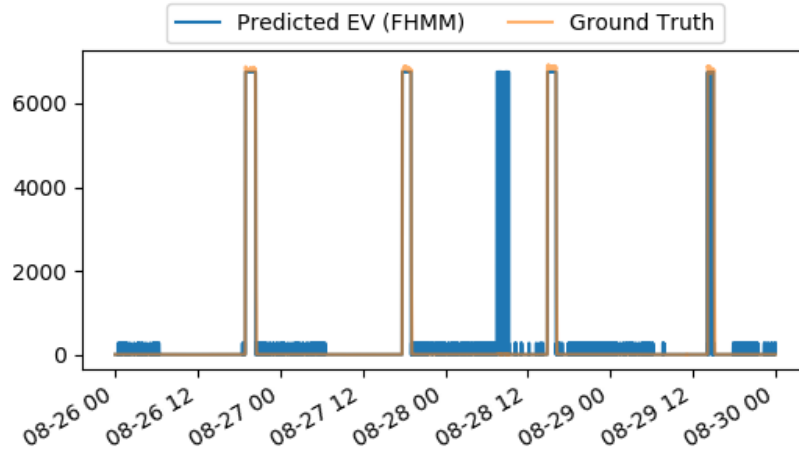


(b) 1-minute, *net\_power*

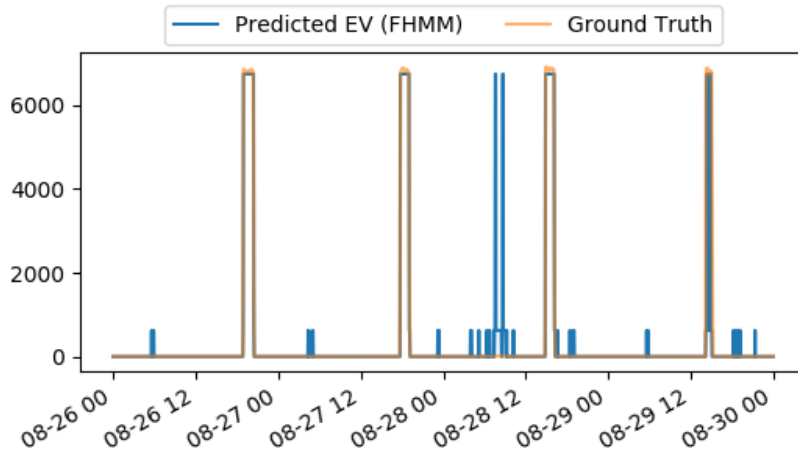


(c) 15-minute, *net\_power*

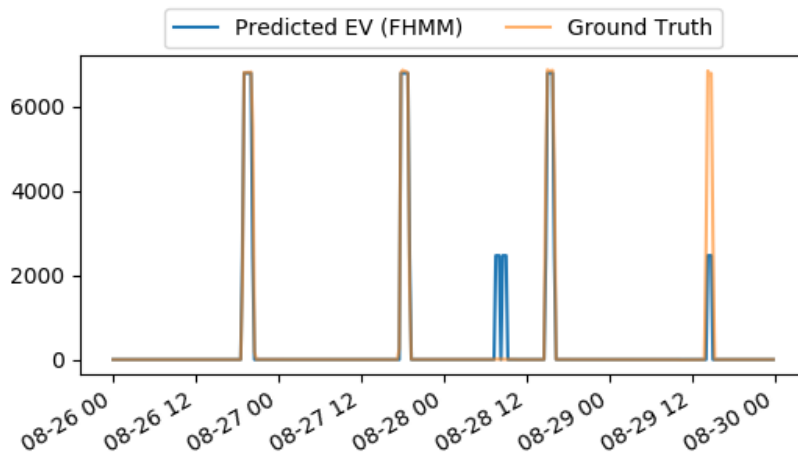
Figure 4.17. The ground truth and predictions of EV charging loads using CO



(a) 1-second, *net\_power*



(b) 1-minute, *net\_power*



(c) 15-minute, *net\_power*

Figure 4.18. The ground truth and predictions of EV charging loads using FHMM

Table 4.7. The output of EV charging profiles based on net\_power.

			Start	End	Elapsed Time	T/F
	GT	1	2019-08-26 18:57:00	2019-08-26 20:25:00	01:28:00	
		2	2019-08-27 17:45:00	2019-08-27 19:02:00	01:17:00	
		3	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00	
		4	2019-08-29 14:10:00	2019-08-29 15:04:00	00:54:00	
1-sec *	CO	1	2019-08-26 18:57:15	2019-08-26 20:25:30	01:28:15	T
		2	2019-08-27 17:45:45	2019-08-27 19:02:15	01:16:30	T
		3	2019-08-28 07:32:15	2019-08-28 08:09:15	00:37:00	F
		4	2019-08-28 08:38:30	2019-08-28 09:13:45	00:35:15	F
		5	2019-08-28 14:52:00	2019-08-28 16:07:30	01:15:30	T
		6	2019-08-29 14:19:15	2019-08-29 14:54:15	00:35:00	T
	FHMM	1	2019-08-26 18:57:15	2019-08-26 20:25:45	01:28:30	T
		2	2019-08-27 17:45:45	2019-08-27 19:02:15	01:16:30	T
		3	2019-08-28 07:32:15	2019-08-28 08:09:15	00:37:00	F
		4	2019-08-28 08:37:30	2019-08-28 09:13:45	00:36:15	F
		5	2019-08-28 14:52:00	2019-08-28 16:07:30	01:15:30	T
		6	2019-08-29 14:19:15	2019-08-29 14:54:15	00:35:00	T
1-min	CO	1	2019-08-26 18:57:00	2019-08-26 20:25:00	01:28:00	T
		2	2019-08-27 17:46:00	2019-08-27 19:01:00	01:15:00	T
		2	2019-08-28 07:32:00	2019-08-28 08:08:00	00:36:00	F
		3	2019-08-28 08:39:00	2019-08-28 08:52:00	00:13:00	F
		5	2019-08-28 14:52:00	2019-08-28 16:06:00	01:14:00	T
		6	2019-08-29 14:19:00	2019-08-29 14:53:00	00:34:00	T
	FHMM	1	2019-08-26 18:57:00	2019-08-26 20:25:00	01:28:00	T
		2	2019-08-27 17:46:00	2019-08-27 19:01:00	01:15:00	T
		3	2019-08-28 07:32:00	2019-08-28 08:08:00	00:36:00	F
		4	2019-08-28 08:38:00	2019-08-28 08:52:00	00:14:00	F
		5	2019-08-28 14:52:00	2019-08-28 16:07:00	01:15:00	T
		6	2019-08-29 14:19:00	2019-08-29 14:53:00	00:34:00	T
15-min	CO	1	2019-08-26 19:00:00	2019-08-26 20:15:00	01:15:00	T
		2	2019-08-27 17:45:00	2019-08-27 18:45:00	01:00:00	T
		3	2019-08-28 07:30:00	2019-08-28 07:30:00	00:00:00	F
		4	2019-08-28 15:00:00	2019-08-28 15:45:00	01:00:00	T
		5	2019-08-28 14:30:00	019-08-28 14:30:00	00:00:00	T
	FHMM	1	2019-08-26 19:00:00	2019-08-26 20:15:00	01:00:00	T
		2	2019-08-27 17:45:00	2019-08-27 18:45:00	01:00:00	T
		3	2019-08-28 15:00:00	2019-08-28 15:45:00	00:45:00	T
		N/A	N/A	N/A	N/A	F

\*The sampling period of 1-second data is 15 due to the lack of computing power on Raspberry Pi.

Figure 4.19 and Figure 4.20 show runtime to train classifiers and disaggregate energy loads. It is seen that FHMM classifier spent significant amount of time as the number of data points increase compared to CO classifier.

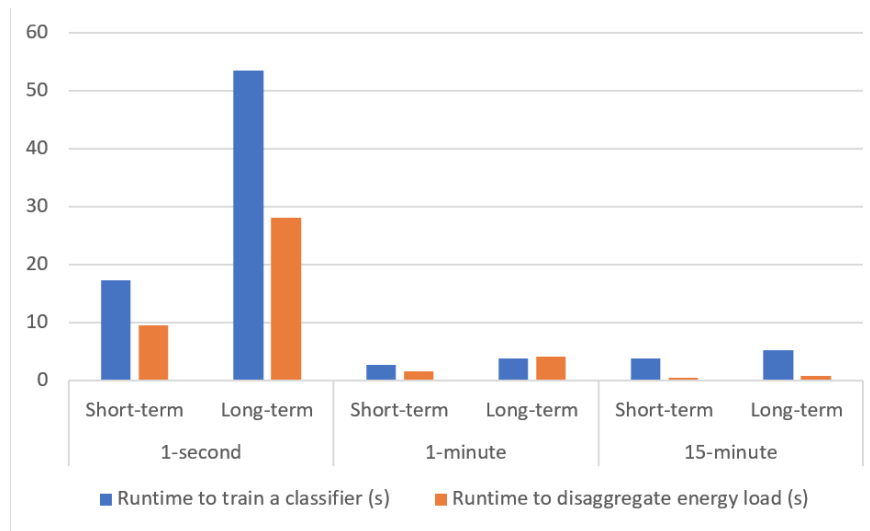


Figure 4.19. Runtime of CO classifier

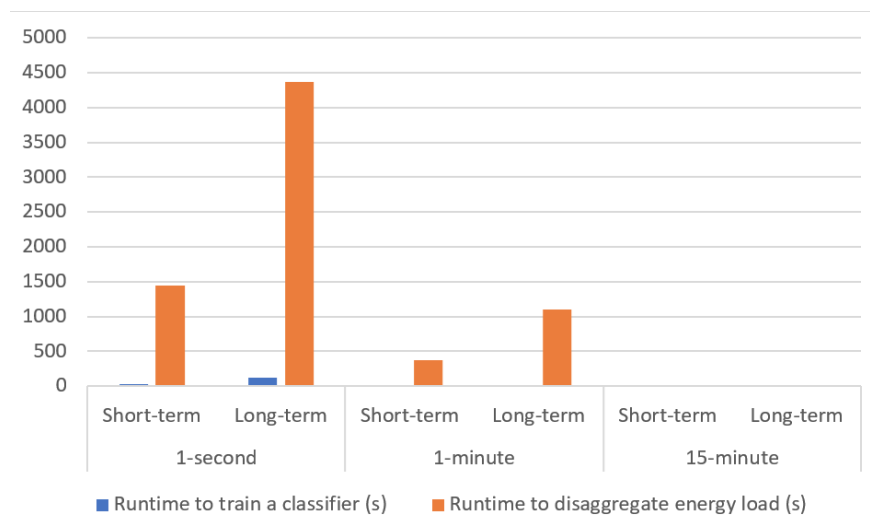


Figure 4.20. Runtime of FHMM classifier

## CHAPTER 5. SUMMARY

This study proposed an implementable end-to-end solution that enables a low-cost edge device to process a massive amount of smart meter data to recognize and identify the electric vehicle (EV) charging loads by utilizing NILM techniques. The process of NILM started with splitting datasets of Dataport, provided by Pecan Street Inc., into several pieces and categorizing files for each house based on assigned house IDs. When the datasets are organized, a converter for recent datasets of Dataport has been implemented in order to employ NILM toolkit (NILMTK) platform. The converter is a new version of the currently provided Dataport converter, aiming to import recent Dataport datasets successfully and reflect solar-generated power when using NILMTK. This task was performed by modifying some functions and metadata. After converting datasets into NILMTK-format, two classifiers-Combinatorial Optimization (CO) and Factorial Hidden Markov Model (FHMM) trained and tested data as NILM algorithms. The experiments were conducted under different conditions: the sampling frequency of data, the number of data points, consideration of solar influx. Finally, the outputs of the experiment were displayed, stored, and exported to a local cloud server. The outputs include Root Mean Squared Error (RMSE) as one of accuracy metrics, plots for ground-truths and predictions of EV charging loads, and a list of information that contains start/end/elapsed time of EV charging loads based on the predictions.

In general, the classifiers trained with *sum\_power* (the total power consumption) outperform those with *net\_power* (the power measured on the grid subtracted by solar-generated power) in both short-term and long-term data experiments. With the presence of *sum\_power*, all the test successfully identified the start and end time of EV charging loads, but the accuracy of elapsed time varies due to the sampling frequency of datasets. As for the tests with *net\_power*, the short-term data experiments showed comparable results with those with *sum\_power*. The predictions in long-term data experiments, however, showed one or two strong non-EV signals along with the real EV signals in every test.

As expected, the CO algorithm had a much faster runtime duration to train classifiers and disaggregate energy data, while FHMM showed better performance with a large amount of data. It must be noted that high sampling frequency data do not guarantee better performance of the NILM algorithms. The tests with higher sampling frequency data generally showed more noise and did not always identify EV loads well, especially for *net\_power* experiments. The tests with lower sampling frequency data showed an almost similar success rate to those with higher sampling frequency data for the number of data points when it comes to generating predictions for EV charging loads. However, the 15-minute frequency data is not always suitable for this study because of the lower accuracy in the elapsed time of the EV loads. Overall, 1-minute frequency data ensures the performance of both classifiers and an extraction algorithm for energy disaggregation.

In conclusion, it is verified that a low-power edge device identifies the electric vehicle (EV) charging load from smart meter datasets using Non-Intrusive Load Monitoring (NILM) algorithms, and further send results to the cloud. This has essential meanings for upcoming smart cities as a low-power, low-cost, tiny edge device can contribute to stable and sustainable smart power grids by providing core data and reducing a substantial amount of costs for sending a whole data. This study has also enhanced the capabilities of NILM approach not only by improving the compatibility between Dataport and NILMTK, but also by extending the study more practical with the consideration of obtained solar-generated power.

Finally, this research leaves a few tasks for future research. First, more algorithms for training datasets will be studied to allow the system to identify EV loads more precisely. Also, future research would work to improve the extraction method to list information about EV charging loads based on predictions correctly. This issue can be resolved by having EV profiles, such as average duration of charge, a maximum peak value of the EV power load. Additionally, along with the currently used data, more data from different household or different state can be compared to evaluate the classifiers and the extraction algorithm.

## REFERENCES

- Anderson, K., Ocneanu, A., Benitez, D., Carlson, D., Rowe, A., & Berges, M. (2012, August). BLUED: a fully labeled public dataset for Event-Based Non-Intrusive load monitoring research. In *Proceedings of the 2nd KDD workshop on data mining applications in sustainability (SustKDD)*. Beijing, China.
- Barker, S., Mishra, A., Irwin, D., Cecchet, E., Shenoy, P., & Albrecht, J. (n.d.). *Smart\*: An open data set and tools for enabling research in sustainable homes*.
- Batra, N. (2015). Non intrusive load monitoring: Systems, metrics and use cases. In *Proceedings of the 13th acm conference on embedded networked sensor systems* (p. 501–502). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/2809695.2822523> doi: 10.1145/2809695.2822523
- Batra, N., Gulati, M., Singh, A., & Srivastava, M. (2013, 11). It's different: Insights into home energy consumption in india.. doi: 10.1145/2528282.2528293
- Batra, N., Kelly, J., Parson, O., Dutta, H., Knottenbelt, W., Rogers, A., ... Srivastava, M. (2014). Nilmtk: An open source toolkit for non-intrusive load monitoring. In *Proceedings of the 5th international conference on future energy systems* (p. 265–276). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/2602044.2602051> doi: 10.1145/2602044.2602051
- Batra, P., Kelly. (2019a). *Currently provided nilmtk converter for dataport*. [https://github.com/nilmtk/nilmtk/blob/master/nilmtk/dataset\\_converters/dataport/download\\_dataport.py](https://github.com/nilmtk/nilmtk/blob/master/nilmtk/dataset_converters/dataport/download_dataport.py). GitHub.
- Batra, P., Kelly. (2019b). *Types of available database supported by nilmtk*. [https://github.com/nilmtk/nilmtk/tree/master/nilmtk/dataset\\_converters](https://github.com/nilmtk/nilmtk/tree/master/nilmtk/dataset_converters). GitHub.
- Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the mcc workshop on mobile cloud computing* (pp. 13–16). New York, NY, USA: ACM. Retrieved from <http://doi.acm.org/10.1145/2342509.2342513> doi: 10.1145/2342509.2342513
- Buneeva, N., & Reinhardt, A. (2017). Ambal: Realistic load signature generation for load disaggregation performance evaluation. In *2017 ieee international conference on smart grid communications (smartgridcomm)* (p. 443–448).



- Cannon, J. (2014). *High availability for the lamp stack: Eliminate single points of failure and increase uptime for your linux, apache, mysql, and php based web applications*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform.
- Chai, T., & Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae) arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3), 1247–1250. Retrieved from <https://www.geosci-model-dev.net/7/1247/2014/> doi: 10.5194/gmd-7-1247-2014
- Chen, S., Wen, H., Wu, J., Lei, W., Hou, W., Liu, W., . . . Jiang, Y. (2019). Internet of things based smart grids supported by intelligent edge computing. *IEEE Access*, 7, 74089-74102. doi: 10.1109/ACCESS.2019.2920488
- Dinesh, C., Welikala, S., Liyanage, Y., Ekanayake, M. P. B., Godaliyadda, R. I., & Ekanayake, J. (2017). Non-intrusive load monitoring under residential solar power influx. *Applied Energy*, 205(C), 1068-1080. Retrieved from <https://ideas.repec.org/a/eee/appene/v205y2017icp1068-1080.html> doi: 10.1016/j.apenergy.2017.0
- Ferrandez, J., Mora, H., Jimeno-Morenilla, A., & Volckaert, B. (2018, 10). Deployment of iot edge and fog computing technologies to develop smart building services. *Sustainability*, 10, 3832:1-23. doi: 10.3390/su10113832
- Gaur, M., Makonin, S., Bajić, I. V., & Majumdar, A. (2019). Performance evaluation of techniques for identifying abnormal energy consumption in buildings. *IEEE Access*, 7, 62721-62733.
- Ghahramani, Z., & Jordan, M. I. (1995). Factorial hidden markov models. *Machine Learning*, 29, 245-273.
- Gron, A. (2017). *Hands-on machine learning with scikit-learn and tensorflow: Concepts, tools, and techniques to build intelligent systems* (1st ed.). O'Reilly Media, Inc.
- Hart, G. W. (1992a). Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), 1870-1891.
- Hart, G. W. (1992b, Dec). Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12), 1870-1891. doi: 10.1109/5.192069

- Kelly, J., Batra, N., Parson, O., Dutta, H., Knottenbelt, W., Rogers, A., ... Srivastava, M. (2014). Nilmtk v0.2: A non-intrusive load monitoring toolkit for large scale data sets: Demo abstract. In *Proceedings of the 1st acm conference on embedded systems for energy-efficient buildings* (p. 182–183). New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/2674061.2675024> doi: 10.1145/2674061.2675024
- Kelly, J., & Knottenbelt, W. (2015a). 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*. New York, NY, USA: Association for Computing Machinery. Retrieved from <https://doi.org/10.1145/2821650.2821672> doi: 10.1145/2821650.2821672
- Kelly, J., & Knottenbelt, W. J. (2015b). The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. In *Scientific data*.
- Khan, L. U., Yaqoob, I., Tran, N. H., Kazmi, S. M. A., Tri, N. D., & Hong, C. S. (2019). Edge computing enabled smart cities: A comprehensive survey. *ArXiv, abs/1909.08747*.
- Kolter, J., & Johnson, M. (2011, 01). Redd: A public data set for energy disaggregation research. *Artif. Intell.*, 25.
- Korte, B., & Vygen, J. (2012). *Combinatorial optimization: Theory and algorithms* (5th ed.). Springer Publishing Company, Incorporated.
- Lai, C., Chien, W., Yang, L. T., & Qiang, W. (2019, April). Lstm and edge computing for big data feature recognition of industrial electrical equipment. *IEEE Transactions on Industrial Informatics*, 15(4), 2469–2477. doi: 10.1109/TII.2019.2892818
- Liu, Y., Yang, C., Jiang, L., Xie, S., & Zhang, Y. (2019, March). Intelligent edge computing for iot-based energy management in smart cities. *IEEE Network*, 33(2), 111–117. doi: 10.1109/MNET.2019.1800254
- Makonin, S., Ellert, B., Bajic, I. V., & Popowich, F. (2016). Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. *Scientific Data*, 3(160037), 1–12.
- Makonin, S., Popowich, F., Bartram, L., Gill, B., & Bajić, I. V. (2013). Ampds: A public dataset for load disaggregation and eco-feedback research. In *2013 ieee electrical power energy conference* (p. 1–6).
- McKinney, W., et al. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th python in science conference* (Vol. 445, pp. 51–56).

- Munshi, A. A., & Mohamed, Y. A. I. (2019, Jan). Unsupervised nonintrusive extraction of electrical vehicle charging load patterns. *IEEE Transactions on Industrial Informatics*, 15(1), 266-279. doi: 10.1109/TII.2018.2806936
- Murugan, D., Garg, A., & Singh, D. (2017). Development of an Adaptive Approach for Precision Agriculture Monitoring with Drone and Satellite Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(12), 5322–5328. doi: 10.1109/JSTARS.2017.2746185
- Nguyen, H.-H. (2020, 01). Nonintrusive load monitoring algorithms: A comparative study. In (p. 212-221). doi: 10.1007/978-981-32-9186-7\_23
- Okay, F. Y., & Ozdemir, S. (2016, May). A fog computing based smart grid model. In *2016 international symposium on networks, computers and communications (isncc)* (p. 1-6). doi: 10.1109/ISNCC.2016.7746062
- Oliphant, T. E. (2006). *A guide to numpy* (Vol. 1). Trelgol Publishing USA.
- Osathanunkul, K., & Osathanunkul, K. (2019). Different sampling rates on neural nilm energy disaggregation. In *2019 joint international conference on digital arts, media and technology with ecti northern section conference on electrical, electronics, computer and telecommunications engineering (ecti damt-ncon)* (p. 318-321).
- Parson, O., Fisher, G., Hersey, A., Batra, N., Kelly, J., Singh, A., ... Rogers, A. (2015). Dataport and nilmtk: A building data set designed for non-intrusive load monitoring. In *2015 ieee global conference on signal and information processing (globalsip)* (p. 210-214).
- Pecan Street Inc. (2019a). Pecan street dataport. Austin, TX, USA. Retrieved from <https://dataport.pecanstreet.org>
- Pecan Street Inc. (2019b). Pecan street dataport. Austin, TX, USA. Retrieved from <https://www.pecanstreet.org>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Plastiras, G., Terzi, M., Kyrkou, C., & Theodoridis, T. (2018, July). Edge intelligence: Challenges and opportunities of near-sensor machine learning applications. In *2018 ieee 29th international conference on application-specific systems, architectures and processors (asap)* (p. 1-7). doi: 10.1109/ASAP.2018.8445118

- Sirojan, T., Phung, T., & Ambikairajah, E. (2017, Dec). Intelligent edge analytics for load identification in smart meters. In *2017 IEEE Innovative Smart Grid Technologies - Asia (ISGT-Asia)* (p. 1-5). doi: 10.1109/ISGT-Asia.2017.8378414
- Syafrudin, M., Fitriyani, N., Alfian, G., & Rhee, J. (2019, 01). An affordable fast early warning system for edge computing in assembly line. *Applied Sciences*, 9, 84. doi: 10.3390/app9010084
- U.S. Energy Information Administration. (2018, 2). Annual energy outlook. In (p. 1-74).
- Wang, S., Du, L. L., Ye, J., & Zhao, D. (2018). Robust identification of ev charging profiles. *2018 IEEE Transportation Electrification Conference and Expo (ITEC)*, 1-6.
- Yang, C., Chen, X., Liu, Y., Zhong, W., & Xie, S. (2019, May). Efficient task offloading and resource allocation for edge computing-based smart grid networks. In *Icc 2019 - 2019 IEEE International Conference on Communications (icc)* (p. 1-6). doi: 10.1109/ICC.2019.8761535
- Yi, S., Li, C., & Li, Q. (2015). A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data* (pp. 37–42). New York, NY, USA: ACM. Retrieved from <http://doi.acm.org/10.1145/2757384.2757397> doi: 10.1145/2757384.2757397
- Zhang, Z., Son, J. H., Li, Y., Trayer, M., Pi, Z., Hwang, D. Y., & Moon, J. K. (2014, Oct). Training-free non-intrusive load monitoring of electric vehicle charging with low sampling rate. In *Iecon 2014 - 40th annual conference of the IEEE industrial electronics society* (p. 5419-5425). doi: 10.1109/IECON.2014.7049328
- Zhao, H., Yan, X., & Ma, L. (2019). Training-free non-intrusive load extracting of residential electric vehicle charging loads. *IEEE Access*, 7, 117044-117053. doi: 10.1109/ACCESS.2019.2936589
- Zhao, H., Yan, X., & Ren, H. (2019). Quantifying flexibility of residential electric vehicle charging loads using non-intrusive load extracting algorithm in demand response. *Sustainable Cities and Society*, 50, 101664. Retrieved from <http://www.sciencedirect.com/science/article/pii/S2210670719306304> doi: <https://doi.org/10.1016/j.scs.2019.101664>
- Zoha, A., Gluhak, A., Imran, M., & Rajasegarar, S. (2012, 12). Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors (Basel, Switzerland)*, 12, 16838-16866. doi: 10.3390/s121216838