Deep Learning on Smart Meter Data: Non-Intrusive Load Monitoring and Stealthy Black-Box Attacks

Junfei Wang, The University of Western Ontario

Supervisor: Pirathayini, Srikantha, The University of Western Ontario
A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering
© Junfei Wang 2020

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Artificial Intelligence and Robotics Commons

Recommended Citation
https://ir.lib.uwo.ca/etd/6891

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlsadmin@uwo.ca.
Abstract

Climate change and environmental concerns are instigating widespread changes in modern electricity sectors due to energy policy initiatives and advances in sustainable technologies. Energy demand management is a strategy that Electric power utility (EPU) companies commonly adopt to modify consumer demand for energy through various method such as time-based rates, incentive-based programs and education. Non-intrusive Load Monitoring (NILM) technique generates appliance-level power consumption data based on a single smart meter readings, providing users more information about the components and proportions of their whole house energy consumption and bills, and creating the data source for Home Energy Management System (HEMS). This thesis proposes a novel deep learning model for NILM that can be used by EPU companies and third party entities for active or passive consumer power demand management. Although machine learning (ML) algorithms are powerful, these remain vulnerable to adversarial attacks. This thesis also studies on the vulnerabilities of ML models in smart grid, and proposes a novel stealthy black-box attack that targets NILM models. Valuable insights are provided for maintaining security especially with increasing proliferation of artificial intelligence in the power system.

Keywords: Non-intrusive Load Monitoring, Deep Learning, Neural Network, Recurrent Neural Network, Adversarial Machine Learning, Power Grid Security, Black-box attack
Summary for Lay Audience

The rapid increase on numbers of smart meters deployment enables utility companies to design and manage various sustainable power consumption programs, such as time-of-use (TOU) and real-time pricing (RTP). Although these programs reduce the engagement of unsustainable and expensive peak-following generation sources, they do not offer so much guidance to improve efficiency of power usage for residents. In this thesis, an ensemble based deep learning model is designed to disaggregated smart meter reading data to appliance-level, so supplying the insight with better granularity. By analyzing and making sense of the disaggregated data, users can engage more into different power usage programs. One potential problem for machine learning based algorithms is they are vulnerable to adversarial attack, which aims to force models making mistakes by only add some indistinguishable perturbations. A novel stealthy algorithm is designed, which can successfully fool deep learning models. It shed lights on the further study to improve the robustness of machine learning model in power grid area.
Acknowledgements

There is so much I want to say at this moment. Pursuing this degree and doing research at Western have never been easy.

My supervisor Dr. Pirathayini Srikantha is the most important person on my research work in Canada. She trusts me a lot and gives me opportunities to trail and error my ideas, although my English speaking was very poor at that time. I even remember how sweaty I was on a weekly meeting. During these two years, I published the first paper in my life, and did the presentation at IEEE conference in Montreal, which are extremely amazing and honorable experiences. Also, the guidance I received benefits me lot both academically and spiritually, that is why I applied to study as her Phd student in the future.

For my family, my wife Mia was pregnant for 9 months when I left them coming to Canada. She is a tough girl! My son, William, I always feel peaceful and full of love when I give him a good night kiss after study at midnight. They are the reason I want to fight.

To my friends Yifang and Mengjie, we came to western at the same term, and always got together at weekends in our first cold winter in Canada. Although we all have families here now and do not meet too often, but I still miss the time when I walked to Yifang’s home on a heavily snowy Sunday morning.

Finally, Just want to tell myself, please believe there is always some hope even when I am down. Please maintain the honesty, faith and consistency on my research.
## Contents

Abstract ................................. i

Summary for Lay Audience .......... ii

Acknowledgements ..................... iii

List of Figures ........................ vii

List of Tables .......................... ix

List of Abbreviations ................. x

1 Introduction .......................... 1

1.1 Overview ............................... 1

1.2 Motivation ............................ 2

1.3 Contributions ....................... 4

1.4 Thesis Organization ............... 4

2 Background ........................... 5

2.1 Non-Intrusive Load Monitoring .. 5

2.2 Feed-forward Neural Network .... 7

2.2.1 Architecture of the Network .... 7

2.2.2 Feedforward Pass ................. 8

2.2.3 Learning Process .................. 8

2.3 Long Short-Term Memory Recurrent Neural Network .......... 10
Vanilla Augmentation Algorithm ........................................... 47
Proposed Data Augmentation Algorithm ................................. 49

5.3 Adversarial Perturbations .............................................. 54
  5.3.1 Fast Gradient Sign Method .......................................... 54
  5.3.2 Proposed Adversarial Perturbation Crafting Algorithm .......... 55
  5.3.3 Performance of Adversarial perturbations ...................... 58
  5.3.4 Impact of Imbalanced Data ........................................ 59
  5.3.5 Stealthiness of Attack Construction ............................... 61

6 Conclusion and Future Work ............................................. 63
  6.1 Conclusion .............................................................. 63
  6.2 Future Work ............................................................ 64

Bibliography ................................................................. 65

Curriculum Vitae ........................................................... 70
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Non-intrusive Load Monitoring by Event Capturing.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>NILM result on ‘Trickl’ of London Hydro Inc.</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>How Perceptrons Are Inspired by Biological Neurons.</td>
<td>15</td>
</tr>
<tr>
<td>2.4</td>
<td>Different Activation Functions for Neural Networks.</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>A Typical FFNN Topology.</td>
<td>16</td>
</tr>
<tr>
<td>2.6</td>
<td>The Architecture of RNN.</td>
<td>17</td>
</tr>
<tr>
<td>2.7</td>
<td>The Structure of LSTM Unit.</td>
<td>18</td>
</tr>
<tr>
<td>2.8</td>
<td>Adversarial Examples on Image Classifier Attack.</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>The Branch Circuit Power Metering for Power Consumption Data Collecting</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Power Consumption Distribution When Cloth Dryer Is On</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Sliding Window Approach</td>
<td>27</td>
</tr>
<tr>
<td>4.4</td>
<td>Power Consumption Percentage of Heat Pump by Quarter</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Minutely Mean Power Consumption Comparison of Cloth Dryer by Hours</td>
<td>29</td>
</tr>
<tr>
<td>4.6</td>
<td>Minutely Mean Power Consumption Comparison of Washer by Day-of-week</td>
<td>29</td>
</tr>
<tr>
<td>4.7</td>
<td>Ensemble Model Architecture</td>
<td>32</td>
</tr>
<tr>
<td>4.8</td>
<td>The Actual Measurement and Disaggregation Result for Dryer</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>Threat Model of Black-box Adversarial Attack</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>Ambiguity in Decision Boundaries</td>
<td>42</td>
</tr>
<tr>
<td>5.3</td>
<td>Transferability Based on Model Complexity</td>
<td>44</td>
</tr>
<tr>
<td>5.4</td>
<td>The Process of Data Augmentation</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>Impact of $\lambda$ on Substitute Model Accuracy</td>
<td>50</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.6</td>
<td>Vanilla vs Proposed Data Augmentation Algorithm</td>
<td>51</td>
</tr>
<tr>
<td>5.7</td>
<td>Impact of $\lambda$ on First-norm of the Normalized Augmented Data</td>
<td>53</td>
</tr>
<tr>
<td>5.8</td>
<td>Conceptual Plot of the Failed Attack Caused by Insufficient Perturbations</td>
<td>56</td>
</tr>
<tr>
<td>5.9</td>
<td>Conceptual Plot of Confidence Margins</td>
<td>57</td>
</tr>
<tr>
<td>5.10</td>
<td>Training Data Visualization in a 2D Space</td>
<td>60</td>
</tr>
<tr>
<td>5.11</td>
<td>Decision Surface of Substitute Model</td>
<td>61</td>
</tr>
<tr>
<td>5.12</td>
<td>The Perturbations over 24 Hours Input</td>
<td>62</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Appliances and Units in AMPds ........................................... 23
4.2 Data Labeling Information ................................................... 26
4.3 Data Imbalance for Appliances in AMPds ................................. 30
4.4 Parameters of RNN ............................................................... 35
4.5 Parameters of FFNN ............................................................... 35
4.6 Performance Comparison ....................................................... 36

5.1 Initial Data set with Simple Examples ..................................... 47
5.2 Evaluation for Substitute Model Training ................................. 54
5.3 Performance When Target Points Are Active ............................. 58
5.4 Performance When Target Points Are Inactive ........................... 59
5.5 Performance Comparison with FGSM ....................................... 59
**List of Abbreviations, Symbols, and Nomenclature**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMI</td>
<td>Advanced Metering Infrastructure</td>
</tr>
<tr>
<td>NILM</td>
<td>Non-intrusive Load Monitoring</td>
</tr>
<tr>
<td>EPU</td>
<td>Electric power utility</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>TOU</td>
<td>Time-of-Use</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Pricing</td>
</tr>
<tr>
<td>AMPds</td>
<td>Almanac of Minutely Power Dataset</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>UAT</td>
<td>Universality Approximation Theorem</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long-short Term Memory</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>FFNN</td>
<td>Feedforward Neural Network</td>
</tr>
<tr>
<td>BCPM</td>
<td>Branch Circuit Power Metering</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>HVAC</td>
<td>Heating, Ventilation and Air Conditioning</td>
</tr>
<tr>
<td>EDA</td>
<td>Exploratory Data Analysis</td>
</tr>
<tr>
<td>BPTT</td>
<td>Back-propagation Through Time</td>
</tr>
<tr>
<td>DAE</td>
<td>Denoising Autoencoder</td>
</tr>
<tr>
<td>HEMS</td>
<td>Home Energy Management Systems</td>
</tr>
<tr>
<td>PGA</td>
<td>Projected Gradient Ascent</td>
</tr>
<tr>
<td>KLD</td>
<td>Kullback–Leibler Divergence</td>
</tr>
<tr>
<td>FGSM</td>
<td>Fast Gradient Sign Method</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Overview

In smart grid, Advanced Metering Infrastructure (AMI) is an integrated system of smart meters that measure the power usage data, communication equipment and networks that transmit the measured data, and data management systems that analyze the data and support policy making. It continuously generates large number of power usage data, enables two-way communication between Electric power utility (EPU) companies and homeowners. Furthermore, EPU companies can utilize these data to reduce supply-demand pressures, and optimize for environmental objectives. Some programs have been widely deployed in practice such as demand response program, which are widely deployed across the residential, commercial and industrial sectors. Time-based utility dynamic pricing schemes such as time-of-use pricing, real-time pricing and critical peak pricing are designed to shave the peak in power demand curve. For EPU companies, these programs may be beneficial because they mainly research on aggregations level of buildings, blocks and regions. However, these policies do not give consumers enough information and guidance on efficient power usage.

Smart meter is the physical component installed on each single building to measure power usage data in AMI. It constantly provides household level data with certain intervals. Non-
Chapter 1. Introduction

Intrusive load monitoring (NILM) is a process for analyzing changes and context in a house and deducing each appliance’s operational state and energy consumption, meaning it transforms household level data to appliance level. Typically, NILM models are trained by EPU companies with a large data set containing power consumption data of various brands and models of appliances, so as to capture general operational patterns of each single appliance. These models will be general models for all users to provide the disaggregated data. With NILM, common issues, like over-estimation or under-estimation of power usage, leaving some appliances at on state forever, using old inefficient appliance which should be replaced, can be visualized in proportions of power consumption and bills. To capitalize these more detailed report and data set, consumers will probably engage more in efficient power consumption by re-scheduling their appliance usage needs, getting rid of old appliances or change their inefficient power usage habits.

Most implementations of NILM is learn patterns of load profile of each appliance with ML models so that accurately provide the states or power consumption value of them. These state-of-the-art models have achieved very high accuracy on NILM problem. However, it is proved that many machine learning (ML) models including deep neural networks are vulnerable to delicately designed adversarial examples[1]. Attackers can affect a target model to generate wrong outputs by only adding small perturbations on the original inputs. This not only happens in the condition that hackers have all the knowledge about the model, but also happens when the knowledge that adversaries hold is very limited like a black-box with only input and output pairs. Therefore, to improve the security and robustness of the power grid while maintaining the intelligence provided by ML algorithms, recognizing the potential vulnerabilities is necessary.

1.2 Motivation

Disaggregated energy usage data are valuable for residents, policy makers, and utility companies. Regarding to residents, reference [2] shows a survey result that only 1 to 2 percent of the
respondents knew how many kWh they used per month or per day and most of them did not even know where their electricity meter was located. This lack of awareness can cause inefficient power usage for the sake of underestimation or overestimation of daily behaviors, using older appliances that burn energy even more costly than purchasing new ones, and leaving appliances at persistently active state. For policy makers, the disaggregated data sets are crucial for developing and evaluating evidence-based energy efficiency policies and programs, so that avoiding additional capacity and capital expenditure. Besides, there are some other features that may influence users’ decision on appliance usage, such as seasons and time. Thus, it is necessary to consider these information as additional features inside a NILM system to further improve the accuracy.

Moreover, utility companies can use the result of NILM to not only display the real-time power usage compositions, or weekly and monthly appliance-level bills to consumers, but also provide seamless customer service and applications with broader strategy. One of the most important applications is appliance scheduling in Home Energy Management Systems (HEMS): in a given time range, the application minimizes the cost of all schedulable appliances while dealing with the leveling of appliances usage demand; meanwhile, it monitors and reacts to the operational states of these appliances and demand tariff. Therefore, if the appliances’ states given by ML-based NILM models in HEMS are incorrect, the appliance scheduling software will react in an inappropriate way. Consequently, adversaries may also design the attacks to drive the HEMS wasting expenses of a building. This risk can be easily exploited by conducting adversarial attacks, because it is proved that ML models have instability to delicately designed adversarial examples although they have exhibited powerful performance in various areas.

From perspectives of game theory, adversarial attack and defense is a zero-sum game, and the goal of defender is to minimize the maximum damage by the attacker. This is actually an arm race between two players.[3] Therefore, discovering more advanced attack mechanisms and identifying potential vulnerabilities of the ML models before they are invented by adver-
saries is a crucial step for improving the robustness and security of smart grid.

1.3 Contributions

As such, main contributions of this thesis are two-fold:

- Firstly, this thesis capitalizes on not only the capability of pattern recognition of deep learning models on time-series data, but also additional features extracted from time stamp which represent users’ general preferences and habits of appliances usage. To learn the two types of data (time-series and additional features), a novel deep learning NILM model is proposed.

- Secondly, as discovering vulnerabilities of proposed ML model provides prior knowledge for building more robust and trustworthy applications. This thesis studies on existing black-box adversarial attack mechanisms and proposes a novel mechanism with Jacobian-based momentum data set augmentation algorithm and gradient based adversarial sample crafting algorithm. Through experiments and testing process, this mechanism proves to be more stealthy and effective to generate adversarial examples for the targeted NILM model.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. The existing work on both NILM and adversarial attack is reviewed and compared in Chapter 3. The background knowledge of these two sub-areas are introduced in Chapter 2, including the context and basic techniques used in the research. The methodology utilized to design the proposed NILM classifier is discussed in Chapter 4 and the adversarial model is illustrated in Chapter 5. Finally, the conclude is made in Chapter 6, and the future research work is also pointed out in this chapter.
Chapter 2

Background

2.1 Non-Intrusive Load Monitoring

In Smart Grid, AMI constantly generates household level power consumption data, but appliance-level data, which have better granularity, are with high demand when house holders or building managers analyze how each single load contribute to the whole building power consumptions and bills. The traditional way to collect these data requires physical installation of sensors on each load across the building, so it is called intrusive load monitoring. Although it can record the exact power usage for each appliance, it is unscalable, inefficient, and expensive.

By contrast, NILM uses software disaggregation algorithm to analyze the patterns inside the whole house data, and accurately outputs the appliance-level power consumption data. This concept was originally invented in reference[4], where the basic process is to detect and monitor the changes on voltage and current, so that identify when different appliances are turned on and off. An example on Fig.2.1[4] illustrated how NILM works: the algorithm tries to capture and recognize individual appliance events from their influence on whole house energy use.

According to the goal of NILM algorithms, they can be classified into three types:

- On/Off Classification: On this level, the NILM model only classifies whether each ap-
pliance is on or off, ignoring the changes during their operations. The advantage is its high accuracy and simplicity. For each appliance, it has only on and off states, they are relatively easier to be captured, so the complexity of models are low. Yet, it treats all running states the same, which may cause overestimation or underestimation of some appliances’ consumptions.

- **Multiple States Recognition:** In some Markov Chain based literature (i.e., [5, 6]), it is called Finite State Machine (FSM). This mechanism categorizes the power consumption of an appliance into different states, trying to recognize which state the target appliance is working on. This type of tasks tries to capture more specific activities when appliances are running, and it also gets rid of details generated by the circuit. The problem is that it is difficult to manually create labels and power consumption ranges of them.

- **Wave Reconstruction:** Its goal is to recover the power wave for each single load. However, it seems impossible to 100% recover every detail of the power signals, because even a same appliance runs several times at the exactly same condition, the circuit may also add random noise onto the waves so that generate different data. Also, the high correlation of appliances will cause serial problems if one random noise or error happens.

In the industry, Enetics, Inc., a US certified meter Data service provider, brought the world’s first NILM product to the marketplace in the Enetics’ SPEED software in 1996 [7]. Since then, 42 EPU and third-party companies around world have provided NILM service to their end-users. Products are for both businesses and households. The purposes include faulty appliance detection, appliance scheduling and customer education. These applications provide users the disaggregated data at different time intervals, such as real-time (secondly), minutely, hourly data. i.e., On the mobile application ‘Trickl’ of London Hydro Inc. (2.2), a Canadian EPU company based on London, Ontario, share the hourly NILM data to customers. According to the survey conducted by Home Energy Analysis, Inc., NILM help saving an average of 12.8% of the energy consumption [8].
2.2 Feed-forward Neural Network

2.2.1 Architecture of the Network

In Artificial Neural Network (ANN), the non-linear computational unit called perceptron is the key component. It is inspired by biological neuron from neuroscience: dendrites receive inputs, and if the inputs are big enough to activate the nucleus, axons will transmit the encoded information to synapses, which take response of transmitting information to other neurons. This process is shown in Fig. 2.3(a)[9]. In ANN, perceptrons imitate the basic operation process of their biological counterparts. This is shown in 2.3(b)[9], where $x_i$ is the information received from the i-th dendrite to the cell body, and $w_i$ is the weight for this input dimension which can be learned through a training process. To generate the output of a neuron, the weighted summation of all inputs is encoded by a non-linear function $f$ called activation function. There are many choices of activation functions, such as sigmoid function, tanh function and rectified linear unit (ReLU) in Fig.2.4[10].

Figure 2.1: Non-intrusive Load Monitoring by Event Capturing.
FFNN is a type of ANNs composed of many perceptrons. In FFNN, these perceptrons are laid out into multiple layers, and there is no loop between layers, meaning the information only flow from previous layers to following layers until reaching the output layer. A typical architecture of FFNN can be shown in Fig. 2.5, in which there are an input layer with three units, an output layer with two units and two hidden layers with many neurons for each. The number of hidden layers and the number of neurons inside each layer are tunable hyper-parameters decided by the complexity of the problem.

2.2.2 Feedforward Pass

Although there exist some connections between ANN and Neuroscience or human brain, it was only inspired by this area to some extent, and the its goal is not to build any artificial neural intelligence or brain function. Rather, it aims to compose many different functions together, so that approximate some complex function $y = f(x)$. Universality Approximation Theorem (UAT) states that a FFNN will be able to approximate any smooth function given that sufficient number of layers and nodes are incorporated into the network[11].

To achieve complex functionality, layers (functions) are connected in a chain to form a composite non-linear function, in which outputs of previous layers are inputs of latter layers. For example, three functions $f_1, f_2, \text{and } f_3$ can represent three layers, and the composite function is in the form of Equation. 2.1, in which $x$ is the input of the network. By adding enough layers and neurons, the complexity of FFNN can be increased.

$$f(x) = f_3(f_2(f_1(x))) \quad (2.1)$$

2.2.3 Learning Process

Neural Networks learn a mapping function from inputs to outputs. This is achieved by updating the weights of the network in response to the errors the model makes on the training data set. To evaluate the errors, a Cost Function, which is used to measure the distance between
model’s output and expected output in the training set, is defined. The common cost function for regression problem is Mean Squared Error (MSE) denoted in Equation 2.2, while in classification problem Cross Entropy (Equation 2.3) is used more often. In both definitions, \( n \) is the number of training examples in a training set, \( y_i \) and \( f(x_i) \) are the expected and actual outputs for the \( i \)-th input of the network respectively. For Cross Entropy, \( k \) is the number of output dimensions.

\[
J = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 \quad (2.2)
\]

\[
J = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_i^j \log(f(x_i)) \quad (2.3)
\]

Although various cost functions are used, they have the common ground: larger cost values denote poor performance, and smaller costs represent better performance. And the goal of the learning process is to keep updating weights inside the model so that decrease the cost. Using Gradient Descent and Backpropagation to learn errors and update all weights regarding to the errors is the classic mechanism to achieve this.

Gradient Descent is the update rule for learning, which is defined in Equation 2.4, in which \( w \) is weight vector in a neural network, \( J \) is the cost function, \( \nabla_w J \) is the first order derivative of \( J \) with respect to each component of \( w \) and \( \alpha \) is learning rate deciding the update step size.

\[
w \leftarrow w - \alpha \nabla_w J \quad (2.4)
\]

However, in most case in multiple layer FFNN, the gradient is impossible to directly calculated. Backpropagation algorithm is an efficient way to achieve this. The ”backwards” part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backwards flow
of the error information allows for efficient computation of the gradient at each layer versus the
naive approach of calculating the gradient of each layer separately.

2.3 Long Short-Term Memory Recurrent Neural Network

RNN adds recurrent connections on FFNN, giving the model another dimension: time. The
architecture is shown on Fig.2.6(a)[12]: with the recurrent connections, input from the previous
time step can be taken into the neuron as part of the incoming information. Fig.2.6(b) illustrates
how the information flows through the network in a feed-forward manner across time can be
visually understood by unrolling the previous diagram in Fig.2.6(a)[12]. One of the most
important properties for RNN is that the weights are shared along all time steps, so the error
would flow back through time when it is trained with Backpropagation algorithm.

However, RNN is known to have the ‘vanishing gradient problem’ when the model is deep.
The errors flowing back through layers tend to decrease to zero, which causes the neural net-
work not learning anything. The most effective way to solve the problem of RNN is to use the
LSTM variant of RNN. The critical component in LSTM is the memory cell, which is shown
on Fig.2.7[12]. The activated block input $z$ calculated by Equation 2.5 is actually a function
of input data and recurrent data from previous time step. The input gate protects the unit from
irrelevant input events, the output $i$ of this gate is derived by Equation 2.6; and the forget gate
helps the unit forget previous memory contents, its output $f$ is obtained by Equation 2.7; Equa-
tion 2.9 is for output gate, which exposes the contents of the memory cell (or not) at the output
of the LSTM unit. The central memory is stored in the ‘Cell’, and its state is calculated by
Equation 2.8, and peepholes provide the state of the ‘Cell’ as additional information to the
three gates. Finally, Equation 2.10 generates the block output of LSTM. The gate structure al-
 lows information to be retained across many time-steps, and consequently also allows gradients
to flow across many time-steps.
2.4 Adversarial Attack

Powerful data-driven ML models have achieved amazing advances these years in many research areas such as computer vision, natural language processing and power grids. Though these state-of-the-art models exhibited excellent performance on classification tasks, reference[13] discovered that there exists a vulnerability in ML models which can be exploited by deliberately crafting some indistinguishable perturbations. This vulnerability reveals attackers can evade human’s detection, and at the same time force ML models making mistakes. Furthermore, the wide use of ML models in many industries motivates adversaries to manipulate the data maliciously.

This type of attack does not influence the training process, and it only attacks at the test phase. It tries to explore how target models work and make decision, so malicious test samples
can be crafted by only adding small perturbations on the original test samples. This attack can be very dangerous in practice. In an autonomous vehicle example in Fig.2.8[14], self-driving cars may recognize a stop sign which is modified with malicious perturbation as a yield sign, so an adversary could potentially use the altered image to cause a car without failsafes to behave dangerously.

Adversaries’ knowledge is a crucial part in a threat model. Reference[15] introduces four dimensions of attackers’ knowledge: the training data $D$; the feature set $X$; the learning algorithm $f$; and the parameters $w$ learned after training the model. According to different levels of knowledge, attacks can be categorized into white-box and black-box scenario.

- **White-box Attack**: The knowledge tuple is $K = (D, X, f, w)$, and this is the best scenario for attackers and worst scenario for model owners, because full knowledge of the target model including model architecture and parameters are exposed to attackers. Hence adversarial samples can be directly crafted on target model. A straightforward method to generate malicious samples is the gradient-based optimization algorithm, it iteratively updates inputs with the following update rule in equation 2.11.

  \[
  \tilde{x} \leftarrow \tilde{x} + \lambda \nabla_{\tilde{x}} J_f(\tilde{x}, \theta) \tag{2.11}
  \]

  It uses gradient descent algorithm on input to perturb and force the cost function keep increasing until the target points is classified as another class.

- **Black-box Attack**: In this scenario, attackers’ knowledge is limited much more than white-box attack, because attackers are assumed only knowing the input feature representation i.e. the dimension $s$ of input, and the meaning of these dimensions) and having access to query the target model. The knowledge tuple is $K = (X)$. A potential solution is to query the model many times to observe how outputs change with different input, but it may take prohibitive times of queries when the target model is complex. It is shown in reference [16] that it takes about 1,500 times of query to solve a logistic regression model
and 11,000 for a simple Neural Network. Furthermore, these numbers all drastically increase while there are more classes or features. However, frequent queries in a short time can very easily expose adversarial activities, and a better attack mechanism is supposed to be stealthy and without much more effort such as brute force attack. In [1], authors proposed a two steps model to efficiently exploit the vulnerability of black-box model, in which it firstly learn a substitute model with Jacobian Data set Augmentation algorithm, and then craft adversarial samples based on this model. This will be discussion in detail in Chapter 5.
Figure 2.2: NILM result on ‘Trickl’ of London Hydro Inc.
2.4. Adversarial Attack

Figure 2.3: How Perceptrons Are Inspired by Biological Neurons.

Figure 2.4: Different Activation Functions for Neural Networks.
Figure 2.5: A Typical FFNN Topology.
2.4. Adversarial Attack

Figure 2.6: The Architecture of RNN.

(a) Showing recurrent connections on hidden nodes

(b) Recurrent Neural Network Unrolled along Time Axis
Figure 2.7: The Structure of LSTM Unit.

Figure 2.8: Adversarial Examples on Image Classifier Attack.
Chapter 3

Literature Review

3.1 Non-Intrusive Load Monitoring

Several types of ML techniques are commonly adopted to solve NILM problem, which take power readings from smart meters as inputs and output disaggregated data. Reference [17, 18, 19] consider NILM problem as a Markov Chain, and try to learn how different states inside the chain transition from each others. Another group of algorithm is Convolutional Neural Network (CNN), which consider smart meter readings as images and scan them to capture patterns of load profiles[20, 21, 22]. Inside the group of CNN, variations are also proposed such as CNN with dilated causal convolutional layers for obtaining broader visual context in reference [20]. Another sub-area of NILM algorithm is Recurrent Neural Network (RNN) [23, 24, 25], which aims to capture the temporal dependencies in power usage sequences. References [26, 27] solve NILM problem with Denoising Autoencoder (DAE): while disaggregated for a specific appliance, power consumptions of other appliances are all considered as noise to be removed.

There exists literature studying on the impact of additional features on power demand and consumption. Reference [28, 29, 30] focus on the subject of energy–weather relationship using primitive variables, such as temperature, relative humidity and solar radiation, and derived variables, including heating degree-days and cooling degree-days.
The research work on NILM in this thesis contains two aspects: firstly, it defines the energy disaggregation task as a time-series data classification problem, and then classify the time-series data with RNN model; secondly, it extracts temporal features from timestamps because it is found in this thesis that there exists energy-time relationships. The combination of the output of the RNN model and the additional features are considered as the input of another neural network. This forms a ensemble model to learn two types of features, which make the study in this thesis differing from all existing literature. Another challenging issue is the imbalanced classification problem which is solved in this thesis via re-sampling techniques.

3.2 Adversarial Attack

An intriguing phenomenon was found in 2014 that several ML models, including state-of-the-art neural networks, are vulnerable to adversarial examples[13]. Reference [31] explained the reason for the existence of adversarial examples as a property of high-dimensional dot products, which is a result of models being too linear, rather than too nonlinear. For black box attack, Reference [16] introduced how an adversary can extract a functional substitute model by collecting input and output pairs. Black-box adversarial attacks on ML models have been introduced by Papernot et al in references [1] in the context of images. The reason why adversarial samples transfer among ML models, and the conditions for these transfers are demonstrated in [15]. The research on adversarial attack in smart grid area is relatively new. Reference [32] directly applies the technique proposed in [31] to manipulate the power data so as to fool the event cause analysis model without being detected by the bad data detectors. Reference [33] introduced a way to fool the event diagnostics model by injecting bad data, and [34] discussed both attack and defense with data injection technique. However, the main study in power grid area seems to simply borrow ideas and algorithms from image recognition and natural language processing, and there is no any related works targeting on NILM classifier. In this thesis, reference [1] is considered as the baseline model. Because it only use one step exploration to
3.2. Adversarial Attack

detect decision boundaries and crafting adversarial examples, it probably results in inaccurate estimation and unnecessary large perturbations. The research work in the thesis considers more complex conditions of black-box attack where the output of the model can be two types, and the decision curve can also be more sophisticated.
Chapter 4

Ensemble-based Deep Learning Model for Non-Intrusive Load Monitoring

Solving NILM on a real data set is challenging, because of the various definition of NILM in different literature, the data imbalanced problem which is unavoidable in real data set, and the accuracy bottlenecks in some cases. In this chapter, section.4.1 firstly introduce the data set, define the problem and then extract useful features from the data set. In section.4.2, an ensemble model with RNN and FFNN is proposed, and its fine-tuned parameters and performance are demonstrated in section.4.3.

4.1 Data Engineering

This section firstly introduces the data set in the research, and then states how the problem is defined. The feature extraction part elaborates the process to construct the two different types of inputs, and finally the imbalanced problem is discussed.
In order to train the NILM classifier, comprehensive power consumption data recorded for each appliance over an extended period of time must be available. As such, the AMPds measurements made available in [35] are utilized. This data set is composed of two years of measurements recorded for the whole house and 20 appliances inside the house. Using branch circuit power metering (BCPM, see Fig.4.1), 21 breakers from the house power panel were metered. The two BCMP units were queried once per minute by an industrial data acquisition server. Table 4.1 lists the 21 sub-metered breakers/loads. The data set provides eleven attributes (e.g. voltage, current, frequency, real power, reactive power and so on) of these appliances at the granularity of one minute, but only the real power measurements provided in this data set are considered since smart meter data is composed of real power readings. This data set is widely utilized for building and testing load monitoring algorithms. Not all 21 meters readings are used as the research targets, rather the loads to be considered are decided based on the Pareto’s 80/20 principle, which means a subset of appliances that consume most of the power is selected as the goal of NILM[18]. As such, the research focuses on six specific appliances out of 21: clothes washer, clothes dryer, dish washer, wall oven, heat pump and HVAC (heating, ventilation and air conditioning)/furnace.
Figure 4.1: The Branch Circuit Power Metering for Power Consumption Data Collecting
4.1.2 State Identification

As introduced in Chapter 2, NILM problem can be classified into three types: On/Off classification, multiple states recognition and wave reconstruction. When an appliance is running, its power consumption may be very different during its operational cycle, so only classifying on/off state cannot capture the accurate power usage at each point of time. By contrast, wave reconstruction aims to recognize all the details inside the aggregated readings, yet this is impossible because even running the same appliance at same condition may generate signals with subtle differences. The trials would probably result in an over-fitting problem, because the model is too sensitive on small changes. Therefore, multiple state recognition is the main goal of the research: modeling the power usage of each appliance into finite states, and then disaggregating the whole house data into one of the states for each of them.

The difficulty for solving a finite states classification problem is the definition of the classes, given only the continuous power consumption data. To extract more states rather than just only off/on state, the histogram of the training set is plotted firstly. Observing and analyzing the histogram helps to make sense of the data distributions, their densities and areas. Then based on these knowledge, the definition of various operating states can be manually carried out. For instance, Fig.4.2 is an example of state labeling for dryers, it is obvious that one state is clustered close to 0 Watts, which reveals the dryer is at the off-state for the most of the time, but when it runs, it has a low power consumption state clustered between about 100 to 300 Watts and a high power consumption state consuming 4,000 Watts and above. Thus, it is evident that there are two dominant states(State 1 and 2) of power consumption in addition to the “off” state (State 0) for the clothes dryer.

Moreover, the number of states and the range of each state in various appliances can be different as well. To extract this information for the six appliances considered in the research work, the histogram of the occurrences of various power consumption levels for each appliances are computed. As such, the dominant states for all six appliances are extracted and these are listed in Table 4.2. This process, re-frames a real value disaggregation problem to be a
finite state classification problem. The mean value of each class is stored during these process, and then the model only output mean value of the target class if real values are needed.

![Power Consumption Distribution When Cloth Dryer Is On](image)

**Figure 4.2: Power Consumption Distribution When Cloth Dryer Is On**

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Labels and Intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloth Washer</td>
<td>S0:0, S1:(0,300], s2:300+</td>
</tr>
<tr>
<td>Cloth Dryer</td>
<td>S0:0, S1:(0,2000], s2:2000+</td>
</tr>
<tr>
<td>Dish Washer</td>
<td>S0:[0,400], s1:400+</td>
</tr>
<tr>
<td>Wall Oven</td>
<td>S0:[0,800], s1:800+</td>
</tr>
<tr>
<td>Heat Pump</td>
<td>S0:[0,500], s1:500+</td>
</tr>
<tr>
<td>HVAC/Furnace</td>
<td>S0:[0,210], s1:210+</td>
</tr>
</tbody>
</table>

**Table 4.2: Data Labeling Information**

4.1.3 Feature Extraction

Based on the research work of this thesis, the power consumption patterns of household appliances is dependent on two types of features: temporal characteristics, which is how the whole house power usage signals change over time; and external characteristics, which is the features and conditions from the environment at the time of monitoring.

To capitalize on the temporal features, the algorithm is designed to scan a piece of memory...
of power usage data. The idea is to solve the current disaggregation problem by looking back into the history. If the memory fragment contains whole history, it would be the most accurate case, because it includes all the details in the past. However, whole history is too long in AMPds (two years), and the length of memories at each monitoring time will be different, which brings more difficulties and inefficiencies on the disaggregation tasks. Therefore, a sliding window approach is adopted to generate fixed length data. The length of this window is sixty minutes which is long enough to contain operational cycles for any appliance in the data set, and the stride of the sliding function is one minute. This is illustrated in Fig. 4.3. Thus, the task of the model becomes: learning the previous one hour aggregated data to classify the state of the target appliance for now.

![Figure 4.3: Sliding Window Approach](image)

The existing literature all focus on introducing different ML models into NILM area so that improve the performance. In another word, the features are fixed, but only applied with different trendy algorithms. However, there are actually other distinguishing features can be utilized into this problem. The exploratory data analysis (EDA) technique, where the main characteristics can be summarized through data visualization, is utilized to extract intelligence from the data set itself. For example, Fig.4.4 shows the percentage of power consumption of heat pump in different quarters, revealing Q1 and Q4 consume about 70% of the total power usage. It is clear that the seasonal influence on the operation of the heat pump. Thus, when the algorithm
encounters any ambiguity for heat pump, seasonal feature can be additional information to
 calibrated the output probabilities.

![Figure 4.4: Power Consumption Percentage of Heat Pump by Quarter](image)

Another example is presented in Fig.4.5 to illustrate how different hours affect the power
 usage: the average power consumption (Watts) of cloth dyer in each hour are very different,
 meaning people tend to use dryers more often during the night, while almost nobody uses
 it in the morning and early afternoon. Similarly, day-of-week is another important feature
denoting people’s preference on using appliances. As shown on Fig.4.6, the usage of washer
 has a dominant feature on Sundays, while the usage on Wednesdays are relatively less often.
 Therefore, three additional features are extracted by EDA process, and these are the month,
 day of week, and hour of the sliding window data which are passed into the classifier as inputs.

### 4.1.4 Data Imbalance and Under-sampling

In ML area, a common problem encountered during training and evaluation is Data Imbalance.
If one of the classes (majority class) inside the data set contains much more examples than other
classes (minority classes), this data set is said to be imbalanced. Although minority classes
are very small, they are often of interest in practical problems. For example, a well-trained
4.1. Data Engineering

Figure 4.5: Minutely Mean Power Consumption Comparison of Cloth Dryer by Hours

Figure 4.6: Minutely Mean Power Consumption Comparison of Washer by Day-of-week
NILM model for Cloth Dryer must have capabilities to capture its running states, although it may only take about one hour out of the whole week. Without this capability, NILM system determines Cloth Dryer is always off, and then the model is worthless. Also, classification accuracy is commonly used as main metrics to measure the performance. If the designer of models has no awareness of the imbalanced issue, it would probably trick the metrics with a extremely high accuracy by only outputting majority class, which means the accuracy is equaled to the percentage of the majority classes. For example, the Cloth Dryer in AMP data set is in off state at 97.9% of the time, and a model directly trained with this imbalanced data will only output off state no matter how different the input is. This model actually does nothing when the dryer runs, but it still has 97.9% accuracy. The percentages of majority and minority classes for each appliance are shown on Table 4.3, revealing that data imbalance exists for every appliance. In order to eliminate the problem of imbalanced classification, the random under-sampling technique is adopted on the data set generated by the sliding window algorithm. The sample rate is decided by the number of samples in minority class, because the aim is to convert the training set to be comparable. It is obvious that some samples would be removed from majority class, but re-sampling will repeat at each epoch of NILM model training, so principally all majority samples will be used when the number of epochs is large enough. For example, consider the clothes washer which is associated with three states: off, washing and spinning that are composed of 1,000,000, 26,000 and 3,000 samples respectively in the data set. Because the minority class spinning only has 3,000 samples, the algorithm under-samples

<table>
<thead>
<tr>
<th>Load</th>
<th>Percentage of Majority Class</th>
<th>Percentage of Minority Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloth Dryer</td>
<td>97.9%</td>
<td>2.1%</td>
</tr>
<tr>
<td>Cloth Washer</td>
<td>97.1%</td>
<td>2.9%</td>
</tr>
<tr>
<td>Dish Washer</td>
<td>98.2%</td>
<td>1.8%</td>
</tr>
<tr>
<td>Furnace/HVAC</td>
<td>99.2%</td>
<td>0.8%</td>
</tr>
<tr>
<td>Wall Oven</td>
<td>99.8%</td>
<td>0.2%</td>
</tr>
<tr>
<td>Heat Pump</td>
<td>92.7%</td>
<td>7.3%</td>
</tr>
<tr>
<td>Average</td>
<td>97.5%</td>
<td>2.5%</td>
</tr>
</tbody>
</table>

Table 4.3: Data Imbalance for Appliances in AMPds
both other two classes to 3,000 too in each epoch. Thus, the training set has 9,000 samples with
3,000 for each state. This technique not only solves the imbalanced classification problem, but
also speeds up the training process, because it is found in the experiments that there are very
high similarities that can be easily classified in most majority classes. This technique is novel
to this thesis.

4.2 Proposed Ensemble-based Deep Learning Model

Given how two types of features are extracted is elaborated in the section 4.1, an ensemble
model to learn both sequential and non-sequential features is designed in this section.

4.2.1 Architecture of the Proposed Model

The proposed solution in this thesis is to build NILM model for each one of the six appliances.
The task of the models is to process the previous 60 minute whole house power consumption
data and the additional temporal features (month, day-of-week and hour) extracted from the
current timestamp so as to label the current operational state of the target appliance.

The proposed model consisting of a RNN and a FFNN is illustrated in Fig.4.7. RNN is se-
lected to read and process the time-series data, the reason is because of its capability of captur-
ing dependencies in sequential data inputs. It was originally created to mimic the learning pro-
cess utilized by humans to understand languages and music from contextual information[36].
Training of weight parameters of the RNN network is conducted using the back-propagation
through time (BPTT) to minimize the errors resulting in temporal learning processes. One
major issue with the standard RNN system is that the vanishing gradient or divergence [37].
To overcome this issue, LSTM units are incorporated. The LSTM units remember the most
‘memorable’ aspects of the data and can process long time series data. In order to increase the
complexity for some appliances, stacked LSTM units are used. This allows for greater abil-
ity to enable abstraction. The output of RNN is the probabilities for the operational states of
the target appliance. i.e. For cloth washer, the output state can be 70% for off state, 20% for washing state, and 10% for spinning state.

Because the month, day-of-week, hour as additional temporal features extracted from timestamp are scalars, so they combine with the probabilistic output of the RNN model as the input of the second model, which is a FFNN. The outputs of the second model are also probabilistic. An intuition for this step is using additional features to improve the accuracy. i.e. For the same case of cloth washer, the output can be improved to 95%, 3%, 2% for the three states if almost nobody uses washer at this time point from history data.

Figure 4.7: Ensemble Model Architecture

4.2.2 Training of the Proposed Model

During the training phase, the data set is divided into training set (80%), validation set(10%) and test set (10%). The training starts at simple model, such at 32 LSTM units and only 1 dense layer with 4 neurons for RNN, and 4 layers FFNN with 64 neurons in total. If a model
is not able to fit the training set well, its complexity should be increased by adding more layers and neurons. To avoid overfitting, in which case the model just remember how training set looks like without capturing general patterns, validation set is used to evaluate the performance of models with different settings. After all these parameters and weights are fine-tuned, the testing set is for final evaluation.

The RNN model and FFNN model in the ensemble model need to be trained separately, these are actually the two steps of the training. At the first step, using the the training set to fine-tune the RNN model, and generate the probability output for all training examples when the model is well-trained. The second step is to train the FFNN, the input of the model is the prediction made by the first model and the additional features.

It is notable that at each epoch of the training, the algorithm will re-sample the training set.

### 4.3 Results

In this section, the performance metrics used in the research are demonstrated firstly, and then the best hyper-parameters of ensemble model for each appliance are tabulated. Finally, the performance of the ensemble model is comprehensively compared with a baseline model.

#### 4.3.1 Performance Metrics

A common choice of performance metrics for a ML classification problem is the accuracy, and it can be defined as in Equation.4.1, in which the correct trials means the number of testing samples that correctly classified by the ensemble model, and total trials is the number of testing samples.

\[
A = \frac{\text{correct trials}}{\text{total trials}} \quad (4.1)
\]

Thus, accuracy is the first metrics in this thesis, and it is useful to reveal the overall performance of the model especially when the testing set is balanced. However, the accuracy metrics will be cheated while encountering imbalanced classification problem. As shown in Table.4.3,
the average percentage of majority class samples is 97.5%, so only measuring accuracy will misunderstand the performance of minority classes.

Hence, F1 score is considered as another metrics, which is the numerous measurement to illustrate how NILM models perform on every state. From classification results, four concept the true positives (TP—positive examples classified as positive), the True Negatives (TN—negative examples classified as negative), the False Positives (FP—negative examples classified as positive) and the False Negative (FN—positive samples classified as negative) can be derived. Then the precision and recall of the classification can be calculated by Equation.4.2 and Equation.4.3.

\[
\text{precision} = \frac{TP}{TP + FP} \tag{4.2}
\]

\[
\text{recall} = \frac{TP}{TP + FN} \tag{4.3}
\]

The F1 score is then calculated to be the harmonic mean of precision and recall:

\[
F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4.4}
\]

For appliances with multiple states, precision and recall are the average of all states [26].

### 4.3.2 Hyperparameters

The parameters of proposed classifier are trained using 80% of the data set in mini-batches with a size of 256. Stacked LSTM is utilized in the models representing the clothes washer, clothes dryer, dishwasher and wall oven for which the distinguishing patterns are more difficult to identify. The parameters utilized in the architecture of models for each appliance are listed in Tables 4.4 and 4.5 for the LSTM RNN and FFNN components respectively. The FFNN component is not involved for the model identifying the furnace operation as non-sequential data does not effect the operation of this load. The parameters listed in these tables have been obtained through multiple trials and design iterations. For the model selection and validation,
10% of the data set is remained.

Table 4.4: Parameters of RNN

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Parameter Type</th>
<th>RNN LSTM Units</th>
<th>RNN Dense Layers Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloth Washer</td>
<td>256,128</td>
<td>64,32,16,8</td>
<td></td>
</tr>
<tr>
<td>Cloth Dryer</td>
<td>256,128</td>
<td>256,128,64,32,16,8</td>
<td></td>
</tr>
<tr>
<td>Dish Washer</td>
<td>256,128</td>
<td>256,512,256,128,64,32,16,8</td>
<td></td>
</tr>
<tr>
<td>Wall Oven</td>
<td>256,256</td>
<td>256,128,64,32,16,8,4</td>
<td></td>
</tr>
<tr>
<td>Heat Pump</td>
<td>128</td>
<td>256,128,64,32,16,8,4</td>
<td></td>
</tr>
<tr>
<td>HVAC/Furnace</td>
<td>64</td>
<td>64,32,16,8,4</td>
<td></td>
</tr>
</tbody>
</table>

*a*numbers split by comma means multiple layers with numbers of neurons.

Table 4.5: Parameters of FFNN

<table>
<thead>
<tr>
<th>Appliance</th>
<th>Layers and Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clothes Washer</td>
<td>128,64,32,16,8</td>
</tr>
<tr>
<td>Clothes Dryer</td>
<td>64,128,256,128,64,32,16,8</td>
</tr>
<tr>
<td>Dishwasher</td>
<td>256,512,256,128,64,32,8</td>
</tr>
<tr>
<td>Wall Oven</td>
<td>64,128,256,128,64,32,8,4</td>
</tr>
<tr>
<td>Heat Pump</td>
<td>64,128,256,128,64,32,8,4</td>
</tr>
<tr>
<td>HVAC/Furnace</td>
<td>-</td>
</tr>
</tbody>
</table>

**4.3.3 Performance and Comparison**

To evaluate the effectiveness of the proposed ensemble-based deep learning model, the comparison is firstly conducted between the model with and without additional features (month, day-of-week and hour). Without additional features, the model is just RNN with LSTM units, its performance is shown in the ‘LSTM RNN’ column in Table.4.6. Experiments show that the ensemble model improve both accuracy and F1 score, as the additional features give the algorithm more information to learn. Especially for Dish Washer and Wall Oven, there are 5 percent increases on accuracy. Therefore, introducing the temporal features results in better accuracy and F1 scores in the ensemble model for all appliances.
Secondly, the comparison is between the ensemble model and a Denoising Autoencoder model (DAE) proposed in reference [26] which is considered as a baseline model. It solves the same problem with the same data set, and provides the F1 score for each appliance. The results of DAE model are listed in the last column of Table 4.6. It is clear that the performance of the proposed model is better than the DAE model.

Table 4.6: Performance Comparison

<table>
<thead>
<tr>
<th>Appliance</th>
<th>LSTM RNN</th>
<th>Ensemble Model</th>
<th>DAE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>F1</td>
<td>F1</td>
</tr>
<tr>
<td>Clothes Dryer</td>
<td>94.40</td>
<td>83.90</td>
<td>94.50</td>
</tr>
<tr>
<td>Dishwasher</td>
<td>86.40</td>
<td>92.54</td>
<td>92.14</td>
</tr>
<tr>
<td>Wall Oven</td>
<td>94.08</td>
<td>96.97</td>
<td>99.34</td>
</tr>
<tr>
<td>Heat Pump</td>
<td>91.56</td>
<td>95.27</td>
<td>93.90</td>
</tr>
<tr>
<td>HVAC/Furnace</td>
<td>92.48</td>
<td>96.45</td>
<td>92.48</td>
</tr>
<tr>
<td>Clothes Washer</td>
<td>89.80</td>
<td>65.80</td>
<td>93.48</td>
</tr>
</tbody>
</table>

In some context of applications, such as history power consumption trends visualization or bill calculation, the time series power consumption data are required being reconstructed, so the system built in the thesis needs to transform the output states back to real value data. As demonstrated in section 4.1, the states of each appliance are manually labeled based on history data, so the mean values of each state are known to EPU companies, and can be stored on their host. When the real value data are required, the system can map the index of a state directly to its mean value.

Fig.4.8(a) shows the measurement of cloth dyer from a physical sub-meter installed in the experimental dryer, and the result of NILM model for dryer is demonstrated in Fig.4.8(b). Comparing the two plots, the proposed model can reconstruct the original signal well.
4.3. Results

Figure 4.8: The Actual Measurement and Disaggregation Result for Dryer
Chapter 5

Adversarial Machine Learning on NILM

With the promising NILM technique being utilized in smart grid, consumers can engage more in sustainable and efficient power usage. By aggregating this improvements of efficiency across houses, communities and areas, unsustainable and expensive peak-following generation sources can be reduced. Furthermore, NILM is building blocks for more complex applications. Particularly, HEMS, which aims to reduce the energy consumption (and more importantly the electricity bill) while maintain occupant’s comfort by scheduling the electrical appliances’ usage[38], is building upon on disaggregated data provided by NILM model. However, ML-based NILM solutions are proved susceptible to adversarial samples, so the potential adversarial attacks targeted this vulnerability may cause NILM models making mistakes, and all applications on disaggregated data will be affected. In this chapter, the vulnerability of the cloud applications built upon on smart metering data is discussed, and some assumptions are made for the adversarial attack. Based on the knowledge of adversaries and the restrictions of the threat model, a novel black-box attack mechanism is proposed. The mechanism adopts the two-step framework from reference [1], with novel Jacobian-based momentum data set augmentation algorithm for substitute training and projected gradient ascent (PGA) with confidence margins to target the NILM model proposed in Chapter 4.
5.1 Threat Model

In this section, the potential system flaws and threats that can be exploited are discussed, and then the insight of the reason for the existences of adversarial samples are introduced. After that, the attack paradigm and adversaries’ capabilities are assumed, and lastly the targeted model in an attacker’s view is illustrated.

5.1.1 Assumptions

It is assumed that the purpose of the attack is to force this NILM model generating wrong labels, so as to confuse the HEMS system of the targeted building and potentially drive it in an costly operation manner. This attack can constantly occur, and a large financial loss would happen to the building owners or the companies. The targeted model utilized in this chapter is the NILM model for HVAC/Furnace proposed in Chapter.4. It provides minutely energy disaggregation result to users, taking 60 minutes smart meter readings, and then deciding whether HVAC/Furnace is active or inactive. The output of this NILM model can be labels of the states (i.e. 0 if inactive or 1 if active) or probabilities (representing the confidence that the appliance is active or inactive). This model will be referred to as the oracle in the reminder of the thesis.

As discussed in Chapter.2, there exist two types of attack paradigms: black-box and white-box based attacks. In this thesis, only black-box paradigm is considered, where the model is trained internally by the solution providers and public access to users or third-party entities in various smart grid applications[39]. The internal components of the ML models are hidden from the public. This implies that an adversary can pose as a third-party entity and query the targeted ML model (i.e. obtain the output for a specific input).

The architecture of the oracle model and its internal parameters are irrelevant for the proposed attack construction, because the black-box properties in this problem. The only information given to adversary will be the dimension of the input \( \vec{x} \) and the type of output \( y \). Thus, the model can be defined as a mapping \( f : \vec{x} \rightarrow y \) where \( \vec{x} \) is the smart meter reading over
60 minutes and the output is either discrete (i.e. active state of each appliance $y \in \{0, 1\}$) or continuous (e.g. probability $y \in [0, 1]$).

It is assumed the adversaries are allowed to query the oracle so as to infer the relationship between input and output. However, to control security challenges via cloud applications, deploying limitation of requests per IP address or user is a common defensive mechanism by solution providers[40]. Therefore, attackers need to consider minimizing the number of queries via the oracle model while designing the adversarial model.

There is no any assumptions for the complexity and types of the targeted ML model, so the adversarial framework proposed in this thesis is a general mechanism for black-box attack. For the output of the oracle model, two types of output (labels and probabilities) from ML models are considered. The scenario that ML models only generate labels is called strict scenario, while the scenario generating probabilities is called slack scenario, because probabilities outputs actually provide more information to attackers.

5.1.2 Vulnerability, Access and Exploitation

In AMI, the power usage data generated by smart meters will be transmitted to EPU companies and stored in cloud server. Nonetheless, this process has malicious intermediary threat, where messages are intercepted and altered with harmful data, and then sent to the destination [41]. Therefore, the threat model is illustrated in Fig.5.1: the target user’s data of the Home Area Network collected by a smart meter are supposed to be sent via data transmission network to AMI host, but attackers intercept the transmission pathway and conduct adversarial attack. One of the characteristics of adversarial samples is its indistinguishability, so it can often evade either human detection or other anomaly detection algorithm. The reason for this considered in the thesis is the inherent ambiguity in decision boundaries between models[42].

In ML classification problems, ML models learn decision boundaries to partition input space into $c$ subsets where $c$ represents the total number of classes. The training set with training examples and labels are collected to provide knowledge of how model designers expect
the model to perform. However, the decision boundaries of ML models are impossible to be the exactly same as the ideal model, so the ambiguities widely exist in ML area. An example can be illustrated in Fig.5.2, there are some blind spots in the non-overlapping areas between the two decision boundaries. Adversaries can move original points into these regions by adding malicious perturbations so that different labels will be generated by two models (ex. moving a to a’, and b to b’ in Fig.5.2). In smart grid and AMI, perturbed smart meter data will evade the anomaly detection process and classified as benign data, but when they are applied on NILM model, specific appliances will be misclassified as incorrect operational state. More severely, other applications such as billing, demand response and data analysis will be devastatingly influenced as a consequence.

5.1.3 Strategy of the Attack

For the nature of black-box attack, the architecture and parameters of the oracle are unknown for adversaries, so the most difficult but essential part of the attack is to obtain the relationship between inputs and outputs of the oracle. Reference[43] states attackers could utilize the same idea of chosen-plaintext attack in Cryptography where attackers design plaintexts and observe the changes of ciphertexts so that reduces the security of the encryption scheme. Thus, ad-
versaries can steal information about how the oracle behaves by strategically design input data and analyze the output data, and then build their own local substitute model. After this step, the black-box attack problem is transformed as a white-box problem with all parameters and relationships available locally. Then the focus of the next step is on how to design high success rate adversarial samples with minimum scales of perturbations. Therefore, the attack strategy in the research contains two steps: firstly building a functionally equivalent local substitute model, and secondly crafting adversarial examples based on this model.

5.2 Substitute Model Construction

5.2.1 Substitute Model Selection

Because of the black-box construction of the problem, the oracle’s architecture is unknown for adversaries. It can be CNN\[20, 21, 22\], RNN\[23, 24, 25\], or Autoencoder\[26, 27\], etc. for a same problem. Therefore, to overcome this issue, the selected architecture of the substitute
5.2. Substitute Model Construction

model should be capable of imitating any type of algorithm inside the black-box oracle. According to the Universality Approximation Theorem (UAT)[11], a FFNN can mathematically approximate any functions when given appropriate parameters. Thus, FFNN is practically a universal choice for Black-box substitute model training tasks. The output of the substitute model is the probability of a class of appliance being active over the interval under consideration and thus is continuous. Once the substitute model is trained to represent the oracle, the internal parameters can be directly used to craft adversarial samples. In other word, substitute model transform a black-box attack to white-box attack. Thus, FFNN is selected to be the internal architecture of the substitute model.

Although the UAT justifies the capability of FFNN, it does not touch upon the learnability of the model[11]. (i.e. the FFNN learns to represent a given model’s, but it does not guarantee to generalize well and produce right prediction on future data.) Thus, the perturbations computed using the substitute model may result in fooling the substitute model but not the Oracle. When a perturbation results in successfully fooling the Oracle, it is referred to as a transferable attack construction. The similarity between substitute model and oracle decides the transferability of attacks. The more similar the substitute model is to the Oracle, the greater will be the transferability.

However, the information of a black-box construction is very limited, so the similarity cannot be directly compared. Reference [15] demonstrates that attack transferability depends strongly on the complexity of the model, and their experiment proves adversarial examples crafted with less complex model would transfer better to the victim model, while adversary examples are more likely stuck at local optimal when they are from a high complexity model. This can be conceptually visualized in an example on Fig.5.3, where Fig.5.3a illustrates the targeted ML model and its decision boundary. Fig.5.3b illustrates the decision boundary computed by a more complex substitute model and Fig.5.3c represents a less complex substitute model. It is clear that the more complex model generate local optima that fools the substitute model but fails to fool oracle, but less complex model transfers better although it travel
further. In general, the complex substitute model fluctuates more often and strongly than the target model, while the sample model generalize the main fluctuations better, and finally find the transferable point. Thus, while picking up hyper-parameters for substitute model, it needs to satisfy two criteria: being complex enough to fit the training set well and being as simple as possible.

![Figure 5.3: Transferability Based on Model Complexity](image)

Next, the cost functions for the strict and slack black-box scenarios have to be different. For
5.2. SUBSTITUTE MODEL CONSTRUCTION

strict scenario, Cross-entropy is selected as the cost function[44] in the form of Equation 5.1,

\[ C = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \]  

(5.1)

where \( N \) denotes the total number of examples in the training set, \( y_i \) is the discrete 0 or 1 output from the Oracle for input \( \vec{x}_i \) (a vector with 60 components) and \( \hat{y}_i \) is the probabilistic output for class \( i \) from the substitute model.

By contrast, in slack scenario, the probability for each state gives adversaries more information about the details of the oracle model. Kullback–Leibler Divergence (KLD) is the cost function chosen to measure the difference between two distributions[45], which matches the goal of this thesis work here. It is defined in Equation 5.2, where \( N \) is the number of examples in the training set, \( y_i \) and \( \hat{y}_i \) are probabilistic output for class \( i \) from the oracle and substitute model. When KLD cost function is minimized, the two distributions are getting closer.

\[ C = \sum_{i=1}^{N} y_i \log\left(\frac{y_i}{\hat{y}_i}\right) \]  

(5.2)

5.2.2 Training Data Augmentation

For the training process of a classifier in ML, a training data set that is full of knowledge on how the trained model is expected to behave is essential. Furthermore, the knowledge is actually means the entropy in Information Theory[46] (i.e. it contribute very small if all training examples are either 100% confident to be class active or inactive; rather it is helpful to have more examples closed to 50%), because training sets with high entropy represent the details of decision boundaries better.

In order to construct this data set, adversaries can collect input/output pairs by querying the oracle model. A brute-force based is introduce in [16], where attackers query targeted model for infinite times, and then use the result collected through this process to train a replica. In fact, however, the number of queries to oracle is monitored and limited, so the data set
collection phase requires a more efficient approach. In this section, a novel iterative synthetic data augmentation algorithm is proposed, and it started from a small data set with simple signals. Jacobian data set augmentation technique is introduce in reference [1] is considered as the baseline for comparison purpose, and it is referred to as the vanilla augmentation algorithm in the remainder of the thesis.

**Initial Data set Collection**

Most literature of adversarial black-box attack focus on image classification problem, where the prior knowledge is so abundant that the initial data set is easy to be collected. For instance, to attack a ML model trained with the MNIST database of handwritten digits[47], one can collect the images of number 1 to number 10 from his or her own handwriting[1] and effectively utilized them as the initial data set for augmentations. However, in aggregated power data analysis, contributions of various appliances and their states can be elusive to adversaries, so the initial data set collection cannot rely on prior knowledge. To bypass this issue, the goal of data augmentation is refined: instead of reproducing synthetic data which are similar to original training examples, data augmentation process aims to iteratively search data points around the oracle’s decision boundary to draw its better representation. Therefore, the initial data set $S_0$ does not have to look like any power signal, they consists of simple data points that include constant power consumption over the 60 minute interval and power consumption that changes over 10 minute intervals. The only condition imposed on these data points is that when these are passed as queries to the Oracle, the outputs must be a balanced representation of various states of each appliance identified by the Oracle, and this can be achieved by several times of trails. In Table 5.1, the initial set of five data points utilized for the appliance class *Furnace* are illustrated, in which each column denotes different 10-minute intervals.
### 5.2. Substitute Model Construction

#### Table 5.1: Initial Data set with Simple Examples

<table>
<thead>
<tr>
<th>id</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>inactive</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>inactive</td>
</tr>
<tr>
<td>3</td>
<td>4500</td>
<td>3500</td>
<td>4500</td>
<td>5500</td>
<td>4500</td>
<td>5500</td>
<td>active</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>500</td>
<td>0</td>
<td>1000</td>
<td>1500</td>
<td>3000</td>
<td>active</td>
</tr>
<tr>
<td>5</td>
<td>2000</td>
<td>10000</td>
<td>8000</td>
<td>1000</td>
<td>1500</td>
<td>7000</td>
<td>active</td>
</tr>
</tbody>
</table>

**Vanilla Augmentation Algorithm**

The vanilla data augmentation algorithm introduced in reference [1] crafts synthetic training inputs by first identifying the directions in which the substitute model’s output is varying and then applying an adjustment along the opposite of these directions to selected data points in the training set. As shown in Fig.5.4, the data set augmentation algorithms utilized for adversarial black box attacks are generally composed of the following steps:

- Identify the output labels for synthetic data points by passing these as inputs to the Oracle;
- Calibrate the substitute model to adjust to the augmented data set
- Iteratively generate new synthetic data points.

These three steps are repeated for several epochs until the synthetic data represent the oracle model well.

The Jacobian matrix $J_f$ of the function $\hat{f}$, where $\hat{f}$ represents the substitute model, contains information about these directions of change and is defined as follows [48]:

$$
J_f = \begin{bmatrix}
\frac{\partial \hat{f}_1}{\partial x_1} & \ldots & \frac{\partial \hat{f}_1}{\partial x_{60}} \\
\vdots & \ddots & \vdots \\
\frac{\partial \hat{f}_m}{\partial x_1} & \ldots & \frac{\partial \hat{f}_m}{\partial x_{60}}
\end{bmatrix}
$$

(5.3)

where the $(i, j)$ entry in $J_f$ is the partial derivative of $\hat{f}$ with respect to the $i^{th}$ component of the input $\vec{x} \in \mathbb{R}^{60}$ and $j^{th}$ output class. The new training sample crafted should represent
the decision boundary of the Oracle. In order to realize this, it is necessary to identify the direction in which the output of the substitute model is least confident (i.e. direction in which the probability of an input belonging to the current class selected by the Oracle is lower). Let $f$ denote the Oracle. The afore-mentioned logic results in the following rule for the vanilla data augmentation technique:

$$S_{\rho+1} \leftarrow \{\bar{x} + \lambda \text{sgn}(J_j[f(\bar{x})]) : \bar{x} \in S_\rho\} \cup S_\rho$$  \hspace{1cm} (5.4)$$

where $S_{\rho+1}$ is the training set that is being currently augmented, $\rho$ denotes the augmentation iteration, $\bar{x}$ is a training point obtained from $S_\rho$, $f(\bar{x})$ is the label obtained from the Oracle for the input $\bar{x}$, $\text{sgn}$ is the function that returns 1 if the input is positive and $-1$ if the input is negative, $J_j[f(\bar{x})]$ is the column of the gradient whose index corresponds to the class the Oracle maps to for input $\bar{x}$ and $\lambda$ is a tunable parameter which alternates between a negative and positive value every 3 iterations that allows for better exploration of the decision boundaries. When the output of the Oracle is probabilistic, a threshold is used to select the class label.
5.2. Substitute Model Construction

(i.e. if the probability is above the threshold, appliance belonging to this class is active and inactive otherwise). This newly synthesized data point is then passed as input to the Oracle in order to obtain the label or confidence of the classes that this point belongs to. After every augmentation, the substitute model is retrained to account for the new point.

The main issue for vanilla data augmentation algorithm is that the exploration of decision boundaries is highly dependent on the initial training data set. According to Equation.5.4, it is a one-step update by adding $\lambda$ or $-\lambda$ as noise on each dimension of the original data points to explore the space. So initial training data set is significant, but actually adversaries have no prior knowledge. Another problem is caused by the imbalanced data set, which does not exist in other literature. The experiment proves small $\lambda$ such as 0.01 drives the algorithm only generating points around the initial data set in a small range, and these points may have same result as the initial data. Algorithm with a large $\lambda$ tends to explore the whole space, but it would be influenced by imbalanced problem, because majority class takes up over 90% of the space in the NILM problem, and finally the substitute model will be trained under a extremely imbalanced synthetic data set, so as to output weak hypothesis. The best $\lambda$ is 0.2, but it still plateau at 78%. The result is shown in Fig.5.5.

Proposed Data Augmentation Algorithm

For the training process of a classifier, training examples that around the decision boundary influence more on the details of the boundary. Given sufficient number of points around decision boundary, the substitute model can be trained to have similar behaviors as the oracle. Vanilla data augmentation algorithm and the proposed data augmentation algorithm share the same ideas: they move current training examples crossing over the decision boundary from currently class to the next class in the current substitute model. Yet, the proposed algorithm in this section is different from the vanilla data augmentation algorithm in reference[1] from three aspects:

- A new data point is not augmented in only one step, so the algorithm does not rely on
the initial data points as much as the vanilla algorithm;

- The update on each dimension has different scale, it gives synthetic data that are closer to the decision boundary;

- To accelerate searching points around the decision boundary and tackle more complex conditions, momentum is considered in addition to the gradient in the augmentation step.

As shown in Fig.5.6, the one step update on the direction to decrease the confidence may result in failing to find satisfied points, because it may misled by the direction of local points. (i.e. In Fig.5.6(a), there is no any points can cross over the 50% confidence boundary on the direction to decrease the confidence of the start point. Multiple steps update with momentum can overcome this issue. Like illustrated in Fig.5.6(b), the algorithm accumulate momentum to overshoot the local optimum and land at the other side of the boundary.

The proposed algorithm is detailed in Alg.1, in which, \( max_p \) is the maximum number of
5.2. SUBSTITUTE MODEL CONSTRUCTION

Figure 5.6: Vanilla vs Proposed Data Augmentation Algorithm

(a) Algorithm that only updates once

(b) Algorithm that travels along the curve with momentum

- Start Point
- On-route Point
- Success Point
Algorithm 1 Substitute model training and data augmentation:

**Input:** $f, \text{max}_\rho, S_0, \lambda, \alpha$

1: Define $\hat{f}$
2: **while** $\rho \in 0 \ldots \text{max}_\rho$ **do**
3: $\hat{D} \leftarrow (\vec{x}, f(\vec{x})) : \vec{x} \in S_\rho$ **» Label the data**
4: train($\hat{f}, \hat{D}$) **» Train the substitute**
5: **for** each $\vec{x} \in S_\rho$ **do**
6: $\hat{y} \leftarrow f(\vec{x})$, $\hat{v} = 0$
7: **while** $\hat{f}(\vec{x}) == \hat{y}$ **do**
8: $\hat{v} \leftarrow \alpha \hat{v} + \lambda \nabla_x J_f[f(\vec{x})]$
9: $\vec{x} \leftarrow \vec{x} - \hat{v}$
10: **end while**
11: $S_\rho \leftarrow \vec{x} \cup S_\rho$
12: **end for**
13: **end while**
14: **return** $S_\rho, \hat{f}$

Points to be augmented into the training set, $\lambda$ and $\alpha$ are parameters that represent step-size and the weight of the momentum, $\alpha$ is typically set to 0.9 to balance the contribution of the gradient term and the momentum, and $\vec{x}$ is a training example contained within the current iteration of the training data set $S_\rho$. At each augmentation iteration, the newly augmented data points in $S_\rho$ are labeled by the Oracle. These new points are utilized to retrain the substitute model $\hat{f}$. Then, in the subsequent search for a new point within the nested while loop, the adjustment to the current data point is iteratively computed using the momentum term $v$ and gradient $J_f[f(\vec{x})]$. It is important to note that in this update, the actual gradient is utilized for the update rather than the sign (e.g. vanilla algorithm). This update is applied in the direction of lower confidence of the updated point belonging to the current class. After the point crosses the decision boundary, the current augmentation iteration ends and the new point is added to the training set. This is repeated until the limiting threshold $\text{max}_\rho$ is achieved (e.g. dictated by the maximum number of Oracle queries to be made by the adversary without detection).

Next, how $\lambda$ affects the first-norm of the normalized augmented data is evaluated as illustrated in Fig. 5.7. When $\lambda$ is large, the augmented data point will be modified significantly and result in faster crossing over the decision boundary due to overshooting. When $\lambda$ is smaller,
although the crossing over occurs less rapidly, the resulting data point will not deviate significantly from the starting point. As evident from Fig. 5.7, the L1-norm deviates significantly from $\lambda = 1.0$ and is more or less similar for $\lambda = 0.1, 0.2,$ and $0.5$. Thus, $\lambda = 0.5$ is selected as it allows for more rapid crossing over across the decision boundary than $\lambda = 0.1$ or $0.2$ with lower impact on the magnitude of the data point.

![Figure 5.7: Impact of $\lambda$ on First-norm of the Normalized Augmented Data](image)

Finally, the performance of the substitute model in both strict and slack scenario is assessed in Table 5.2. To reflect the overall performance of the models, the test points are selected to be balanced (50% are active and another 50% is inactive). The results show that both of the two scenarios perform well with more than 90% accuracy, it is much better than vanilla data augmentation algorithm which has 78% accuracy in the best condition. Twice the queries were needed for the slack scenario because it contains more

The accuracy of the substitute is measured with respect to the output of the Oracle (i.e. percentage of outputs from the substitute that match the Oracle). It is clear that both types of
Oracle outputs (i.e. discrete and probabilistic) perform well with more than 90% accuracy although twice the queries were necessary for the probabilistic output that contains more nuanced information.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Epochs</th>
<th>Queries</th>
<th>Accuracy toward Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strict</td>
<td>7</td>
<td>1210</td>
<td>92.99%</td>
</tr>
<tr>
<td>Slack</td>
<td>8</td>
<td>2174</td>
<td>90.79%</td>
</tr>
</tbody>
</table>

Table 5.2: Evaluation for Substitute Model Training

5.3 Adversarial Perturbations

The substitute model will be utilized to craft adversarial perturbations to input features that result in the assignment of incorrect labels by the Oracle. The main objective in the design of these perturbations is to fool the oracle model and evade the error checking or anomaly detecting mechanisms that validate the smart meter measurements. Thus, the magnitude of these perturbations must be negligible so that these are not detectable. Mathematically, this problem amounts to the Equation.5.5, where $\delta_x$ is the adversarial perturbation applied to the original input $x$, and $r$ is the limitation on the infinite norm of the change.

$$f(x) \neq f(x + \delta_x)$$

$$s.t. ||\delta_x||_\infty < r$$

5.3.1 Fast Gradient Sign Method

Reference [1] proposed an algorithm called fast gradient sign method (FGSM), and become widely used in the literature. It uses the sign of the gradient of the cost function $C$ of the substitute model $\hat{f}$ to update the original samples so as to search adversarial examples. The perturbation is calculated by Equation.5.6[1], where $\lambda$ is the parameter that is tried in the algorithm so that the label produced by the substitute model for the perturbed input data (i.e.
$\vec{x} + \delta_{\vec{x}}$ changes from the original label to a different one and fools the Oracle.

$$\delta_{\vec{x}} = \lambda \text{sgn}(\nabla_{\vec{x}} \hat{f})$$ (5.6)

Although this is a very straightforward method, there exist three main problems with this approach for the NILM problem. One is that the magnitude of perturbation applied to each component or feature of $\vec{x}$ will be the same. Thus, even if one component of $\vec{x}$ need not be perturbed as much as the other component, it will experience the same magnitude of perturbation which can result in detection. The second issue is with the parameter $\lambda$ which can be increased until the substitute model misclassifies the perturbed point resulting in easy detection of the attack. Thirdly, as there is no constraint imposed on $\lambda$, these perturbations can result in infeasible outputs (e.g. negative power readings) which can be easily detected.

### 5.3.2 Proposed Adversarial Perturbation Crafting Algorithm

The issues of FGSM algorithm are solved by proposing a PGA method that defines a $\ell_{\infty}$-ball and projects perturbed $\vec{x}$ into this legitimate space[49].

The $\ell_{\infty}$ norm of a vector is defined to be the maximum component. The radius of the $\ell_{\infty}$-ball is constrained to be within a specific threshold $r$ which varies for each targeted data point as it is selected to be a percentage $p$ of the mean value of the input signal. Input perturbations are applied iteratively so that these remain within the boundaries defined by the $\ell_{\infty}$ ball while moving in the direction that increases the cost incurred by the substitute model as follows:

$$\vec{x} \leftarrow f_p(\vec{x} + \lambda \nabla_{\vec{x}} \hat{f}, r)$$ (5.7)

where $f_p$ projects the perturbed $\vec{x}$ into the $\ell_{\infty}$-ball. The perturbation applied to each component of $\vec{x}$ varies based on the value taken by the gradient of the cost function $C$. The gradient can now be exactly calculated as the attacker has access to the internal parameters and architecture of the substitute model. These updates are repeated until the output of the substitute model changes.
from the original output to another output class or the algorithm exceeds the maximum limit $n$ of iterations.

In reality, the substitute model is not an exact copy of the Oracle. In many cases, adversarial samples fool the substitute but fail to transfer to oracle model. For example, in Fig. 5.8, a’ is the point generated by using PGA. It is classified to be class 2 by the substitute model, but still belongs to class 1 in oracle model. So if it travels a little further to the position of a”, it can successfully transfer. In order to ensure that the effect of the perturbations applied to the substitute model transfers over to the Oracle, the notion of confidence margin $m$ is introduced where the probability of the dominant output class of the original unperturbed input is higher.

![Figure 5.8: Conceptual Plot of the Failed Attack Caused by Insufficient Perturbations](image)
5.3. **Adversarial Perturbations**

than that of the dominant output class of the perturbed input by $m$.

The margin can be conceptually drawn in Fig.5.9: the confidence margin is a parameter to be tuned, the algorithm have to move target points across the margin to be adversarial examples. This process avoid adversarial examples stuck at local optimum of substitute, but the price is

![Figure 5.9: Conceptual Plot of Confidence Margins](image)

that the more confident it is the more perturbations will be added to the data. When constructing the perturbations, this confidence margin is maintained according to Alg. 2.
Algorithm 2 Adversarial Perturbation Crafting:

Input: \( \hat{f}, \lambda, C, m, n, p \), target example \( \vec{x} \)

1: \( \hat{y} \leftarrow \hat{f}(\vec{x}) \) \>
   Save the original result
2: \( r = pE[\vec{x}] \) \>
   Calculate radius of \( \ell_\infty \)-ball
3: repeat \( n \) times
4: \( \vec{x} \leftarrow f_p(\vec{x} + \lambda \nabla_x C(f(\vec{x})), r) \)
5: until \( \argmax \hat{f}(\vec{x}) \neq \argmax(\hat{y}), \max \hat{f}(\vec{x}) - \max \hat{y} \geq m \)
6: return \( \vec{x} \)

### 5.3.3 Performance of Adversarial perturbations

In Tables 5.3 and 5.4, percentages of adversarial inputs that succeed in fooling the Oracle (i.e. transfer rate) for various thresholds and confidence margins are tabulated for the active and inactive states of a furnace. As expected, higher values of \( r \) permits greater success rates in crafting adversarial examples and higher confidence margins result in greater transferability between the substitute and Oracle models. The difference in the success and transfer rates for the active and inactive states are due to the distribution of the data points. For instance, the active-state is defined by input points that are concentrated and the inactive-state are defined by points that are more dispersed. Thus, smaller perturbations are needed to transition from the active to inactive state but not vice versa. If the initial training points are closer to the class boundaries, then small perturbations that heed \( \ell - \infty \) boundaries will be sufficient for the attack.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Success Rate</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>r=10% mean</td>
<td>15.2%</td>
<td>14.3%</td>
</tr>
<tr>
<td>r=20% mean</td>
<td>37.5%</td>
<td>13%</td>
</tr>
<tr>
<td>r=50% mean</td>
<td>80.4%</td>
<td>45.9%</td>
</tr>
<tr>
<td>r=50% mean+10% margin</td>
<td>79.3%</td>
<td>52.1%</td>
</tr>
<tr>
<td>r=50% mean+20% margin</td>
<td>79.3%</td>
<td>52.7%</td>
</tr>
<tr>
<td>r=50% mean+50% margin</td>
<td>75.5%</td>
<td>66.2%</td>
</tr>
<tr>
<td>r=50% mean+80% margin</td>
<td>57.6%</td>
<td>89.6%</td>
</tr>
</tbody>
</table>

Table 5.3: Performance When Target Points Are Active

In Table 5.5, the performance of the proposed PGA algorithm with 100% mean value radius and 20% confidence margin and the FGSM algorithm on success rate of searching (i.e. percentage of adversarial examples found from the test set) and the transfer rate (i.e. percentage
5.3. Adversarial Perturbations

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Success Rate</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>r=50% mean</td>
<td>12.1%</td>
<td>23.5%</td>
</tr>
<tr>
<td>r=80% mean</td>
<td>24.3%</td>
<td>20.6%</td>
</tr>
<tr>
<td>r=100% mean</td>
<td>31.4%</td>
<td>18.2%</td>
</tr>
<tr>
<td>r=100% mean+20% margin</td>
<td>28.6%</td>
<td>22.5%</td>
</tr>
<tr>
<td>r=100% mean+50% margin</td>
<td>22.9%</td>
<td>25%</td>
</tr>
</tbody>
</table>

Table 5.4: Performance When Target Points Are Inactive

of adversarial examples that successfully fool the Oracle model).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type</th>
<th>Success Rate</th>
<th>Transfer Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGSM</td>
<td>Discrete</td>
<td>100%</td>
<td>2.6%</td>
</tr>
<tr>
<td></td>
<td>Probabilistic</td>
<td>97%</td>
<td>4.7%</td>
</tr>
<tr>
<td>PGA</td>
<td>Discrete</td>
<td>48.6%</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>Probabilistic</td>
<td>54%</td>
<td>37.6%</td>
</tr>
</tbody>
</table>

Table 5.5: Performance Comparison with FGSM.

It is clear that the proposed algorithm is more successful in transferring the attack construction to the Oracle model than reference [1] on both discrete and probabilistic scenarios, although FGSM algorithm can almost find a adversarial point for each test point for the sake of no restriction.

5.3.4 Impact of Imbalanced Data

As the results of adversarial attacks shown in Table.5.3 and Table.5.4, the success rate and transfer rate on active and inactive class data are very different. This is mainly because of the imbalanced distribution of the original training data. In this section, data visualization and decision boundary visualization are utilized to help analyzing how imbalanced data set influence adversarial attacks.

Training data distribution influences classifiers trained on it fundamentally, because the training process is to gradually understand and represent the training data. The scatter plot of the original training data is shown in Fig.5.10, where the 60 dimensional data is mapped on to a 2 dimensional plane with Principal Component Analysis technique. It is clear that active-state
points cluster in a relatively small range, while inactive-state points are almost everywhere inside the space. This imbalance of distribution affects the shape and position of the decision boundary.

Fig. 5.11 shows the decision surface of the substitute model in a 3 dimensional space, where the principal component 1 and principal component 2 are two random variables representing the power consumption signal, and the vertical axis is the probability denoting how confident the testing points belonging to active-state. Active-state region is at the upper-left corner, while the remain of the space is all inactive. This explains why active-state points are easier to escape their small region to be inactive, but inactive points may at any position on the blue area which is far from the decision boundary.

Furthermore, active-state points commonly have larger mean value so the larger $r$ in Alg. 2. This allows them perturbing more on the targeted points.

![Training Data Visualization in a 2D Space](image)
5.3. Adversarial Perturbations

5.3.5 Stealthiness of Attack Construction

Next, the stealthiness of the attack construction is presented. One method of establishing this is via visual inspection. As such, Fig. 5.12 illustrates the smart meter reading for a household over a 24 hour period where perturbation is applied to a single one hour window highlighted by the orange curve. It is clear that the perturbed data is not distinguishable from actual smart meter readings. With this attack, smart meter readings can be modified in a manner that is not noticeable to the EPU and result in over-billing (e.g. consumer extortion) or under-billing (e.g. energy theft) for specific use of particular appliances.
Figure 5.12: The Perturbations over 24 Hours Input
Chapter 6

Conclusion and Future Work

6.1 Conclusion

In conclusion, this thesis mainly discusses two applications building on smart meter data. The first application is ML-based NILM, which aims to decompose the whole house power consumption reading into appliance level measurements. The task is defined as a multiple class classification problem for each appliance, as the running states of each of them are labeled during data pre-processing phase. The smart meter readings are naturally time-series data, while new features month, day-of-week and hour are scalars selected to help improving the models’ performance. To process these two types of input data, an ensemble-based deep learning model is proposed. Models in existing literature (RNN and DAE) are comprehensively compared using accuracy and F1 score metrics. This work can be readily further trained and deployed onto EPU companies’ applications to incentivize sustainable and economical power usage.

The second application is related to cyber security. Because the result of NILM provide prior knowledge for HEMS especially appliance scheduling applications, the robustness of NILM models is important. To discover new adversarial algorithms and new vulnerabilities before hackers helps building more trustworthy ML models in smart grid, so the second application focus on adversarial attack. It is proved that ML models are vulnerable to adversarial samples
that are deliberately crafted to force the models generating incorrect labels, but the existing algorithms are struggled to overcome classifiers trained on imbalanced data set. Therefore, a novel black-box attack mechanism with a new local substitute model training algorithm and a novel adversarial sample crafting algorithm is proposed in this thesis. By comparing the performance with existing algorithm, this new mechanism performs better.

6.2 Future Work

As future work, firstly, the data set AMPds used in this thesis is only based on one house, it may not represent how appliances of different made or working status, and the conditions that several same appliances working together at the same time (e.x. two washers operating simultaneously), and a group of people’s habits in general. These problems are tough but practical, these are worth investigating with the real life data. Thus, the next step for NILM research is to collect more comprehensive data set and build model that can capture more general patterns. Secondly, the outputs of different models may have complex correlations, and these knowledge may be helpful to reduce ambiguities about states. This point means the output of NILM for each appliance can be combined and considered together, so that form a collaborating system. Another interesting topic is appliance scheduling building upon on NILM data and the EPU company’s price schema. It is highly related to Reinforcement Learning (RL), where an agent tries to learn the environment so that achieve the most reward. It can be seen as a scheduling robot learning to operate appliances as requested by the house owner to achieve lowest bill while maintain the comfort at the same time. Fourthly, a next step of adversarial attack is the defensive mechanism in smart grid. The key for this problem is to reduce the ambiguities and blind spots of the oracle’s decision boundary, but do not decrease its performance.
Bibliography


Curriculum Vitae

Name: Junfei Wang

Post-Secondary Education and Degrees:
Beijing Institute of Petrochemical Technology
Beijing, China
2004 - 2008 B.E. on Communication

Beijing University of Posts and Telecommunications
Beijing, China
2008 - 2010 M.Eng., Software Engineering

University of Western Ontario
London, ON, Canada
2018 - 2020 M.E.Sc., Electrical and Computer Engineering

Publications:


J.Wang, P.Srikantha. Stealthy Black-box Attacks on Smart Meter Based Deep Learning Models. IEEE Transactions on Smart Grid. (Being under Review)