



LEARNING ALGORITHMS FOR NON INTRUSIVE LOAD MONITORING

BY

SHIKHA SINGH

Under the Supervision of Dr Angshul Majumdar

Indraprastha Institute of Information Technology Delhi

July, 2022

©Indraprastha Institute of Information Technology (IIITD), New Delhi, 2021



LEARNING ALGORITHMS FOR NON INTRUSIVE LOAD MONITORING

BY SHIKHA SINGH

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

**Doctor of Philosophy**

ELECTRONICS AND COMMUNICATION ENGINEERING

INDRAPRASTHA INSTITUTE OF INFORMATION TECHNOLOGY - DELHI

NEW DELHI- 110020

JULY, 2022

# Certificate

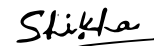
This is to certify that the thesis titled *Learning Algorithms for Non Intrusive Load Monitoring* being submitted by *Shikha Singh* to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of Doctor of Philosophy, is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standard fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree or diploma.



July, 2022

Dr. Angshul Majumdar



Indraprastha Institute of Information Technology Delhi

New Delhi 110020



# Abstract

Non-Intrusive Load Monitoring (NILM), also known as Energy Disaggregation, is the process of segregating a building's total electricity consumption into its appliances by appliance consumption from the smart meter data. In developed countries, Smart meters are currently being rolled out on a large scale, mainly due to the two most important benefits of energy disaggregation: 1)helps consumers understand their energy usage by providing itemized bills; 2)helps grids with capacity planning. These benefits ultimately lead to energy saving and cost-cutting.

Existing algorithms for NILM consist of a training phase in which sub-metered appliance data is used to build models of the appliances. In the test phase, these models are used to disaggregate the total electrical energy consumption. To collect the appliance-level data, we need to put sensors on appliances present in the building. This makes the training phase intrusive. Due to this, such methods do not provide a scalable solution for NILM.

This thesis has three main objectives: 1)To propose more accurate algorithms for NILM than state of the art; 2)To propose the methods which make the training phase completely non-intrusive, i.e., to dodge the requirement of the sub-metered data ; 3)To propose an algorithm that can work with compressed energy signals in order to save the bandwidth and avoid network congestion.

First, we propose a dictionary-learning based algorithm called Deep Sparse Coding for NILM. The usual technique is to learn a dictionary for every device and use the learned dictionaries as a basis for blind source separation during disaggregation. Prior studies in this area are shallow learning techniques, i.e., they learn a single layer of dictionary for every device. In this work, we learn multiple layers of dictionaries for each device. These multi-level dictionaries are used as a basis for source separation during disaggregation. We show that this algorithm outperforms the benchmark techniques like Factorial Hidden Markov

## Model and Discriminative Disaggregating Sparse Coding.

Second, we follow the multi-label classification based paradigm for NILM and determine the state(On/Off) of the appliances present in the building. For this, we propose several algorithms that adapted Transform Learning, Sparse Representation Classifier, Restricted Boltzmann Machine and Long Short Term Memory to perform multi-label classification and subsequently disaggregating the appliance-level load.

Third, we propose a compressed sampling(CS) approach. The high-frequency power signal from a smart meter is encoded (by a random matrix) to very few samples making the signal suitable for WAN transmission without choking network bandwidth. CS guarantees the recovery of the high-frequency signal from the few transmitted samples under certain conditions. This work shows how to recover the signal and simultaneously disaggregate it.

The motive of the work presented in this thesis is to propose NILM algorithms independent of the sub-metered data and make advancements in state-of-the-art in the field of energy disaggregation.

# Dedication

To,  
My loving grandparents

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Dr Angshul Majumdar for his constant support, patience and motivation in my research journey so far. His guidance helped me throughout the time of my PhD and writing of this dissertation. I would like to thank him for providing outstanding research environment and facilities.

I would like to thank my committee members Dr Shobha Sundar Ram and Dr P. B. Sujit. for their insightful comments. I am grateful to the Indraprastha Institute of Information Technology for providing an excellent infrastructure.

I thank my labmates, namely Vanika Singhal, Megha Gupta Gaur, Jyoti Maggu, Aanchal Mongia, Shalini Sharma, Pooja Gupta and Anurag Goel, for their companionship and never-ending tea-time stories. I gratefully acknowledge their help and support.

I would like to thank my family, especially my father and mother, for always believing in me, for their continuous love and support throughout my life. I owe a special thanks to my husband, Alok Singh, for his care, patience, encouragement and unwavering support throughout this journey.



# Contents

<b>Abstract</b>	<b>i</b>
<b>Dedication</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Problem Statement . . . . .	3
1.2 Background . . . . .	4
1.2.1 Event-based Methods . . . . .	5
1.2.2 Finite State Machines . . . . .	6
1.2.3 Sparse Coding . . . . .	7
1.2.4 Discriminative Disaggregating Sparse Coding . . . . .	9
1.2.5 Neural Networks . . . . .	11
1.2.6 Multi-Label Classification Based Approaches . . . . .	12
1.3 Datasets . . . . .	13
1.3.1 Reference Energy Disaggregating Dataset . . . . .	13

1.3.2	Pecan Street Dataset . . . . .	14
1.4	Research Contributions . . . . .	14
<b>2</b>	<b>Deep Sparse Coding and Analysis Co-Sparse Coding for Non-Intrusive Load Monitoring</b>	<b>16</b>
2.1	Literature Review . . . . .	17
2.1.1	Synthesis Sparse Coding . . . . .	17
2.1.2	Analysis Co-Sparse Coding . . . . .	18
2.2	Proposed Formulation for Deep Sparse Coding . . . . .	20
2.2.1	Greedy Solution . . . . .	23
2.2.2	Exact Solution . . . . .	25
2.2.3	Energy Disaggregation . . . . .	27
2.3	Proposed Formulation for Analysis Co-Sparse Coding . . . . .	28
2.3.1	Simple Co-Sparse Coding . . . . .	29
2.3.2	Distinctive Dictionaries . . . . .	34
2.3.3	Disaggregating Dictionaries . . . . .	35
2.4	Experimental Evaluation . . . . .	38
2.4.1	Results with Deep Sparse Coding . . . . .	40
2.4.2	Results with Analysis Co-Sparse Coding . . . . .	49
2.5	Discussion . . . . .	53
<b>3</b>	<b>Non-Intrusive Load Monitoring via Multi-Label Classification</b>	<b>55</b>
3.1	Literature Review . . . . .	56
3.2	Proposed Formulations . . . . .	58
3.2.1	Multi-Label Sparse Representation-Based Classification	58
3.2.2	Multi-Label Restricted Boltzmann Machine . . . . .	59
3.2.3	Multi-Label Deep Convolutional Transform Learning . .	63

3.3	Experimental Evaluation . . . . .	66
3.3.1	Results- Multi-Label Sparse Representation-Based Classification . . . . .	68
3.3.2	Results- Multi-Label Restricted Boltzmann Machine . . . . .	69
3.3.3	Multi-Label Deep Convolutional Transform Learning . . . . .	71
3.3.4	Classification . . . . .	71
3.3.5	Regression . . . . .	72
3.4	Discussion . . . . .	73
<b>4</b>	<b>Blind Compressed Sensing for Non-Intrusive Load Monitoring</b>	<b>75</b>
4.1	Literature Review . . . . .	76
4.1.1	Blind Compressed Sensing . . . . .	77
4.2	Proposed Formulation for Blind compressed NILM . . . . .	81
4.2.1	Training . . . . .	81
4.2.2	Testing . . . . .	83
4.3	Proposed formulation for Deep Blind Compressed NILM . . . . .	85
4.4	Experimental Evaluation of BCS . . . . .	87
4.5	Experimental Evaluation of Multi-Label BCS . . . . .	88
4.6	Discussion . . . . .	92
<b>5</b>	<b>Conclusion</b>	<b>93</b>
5.1	Summary of Contribution . . . . .	93
5.2	Future Work . . . . .	94
5.2.1	Domain Adaptation . . . . .	95
5.2.2	Training-less Non-Intrusive Load Monitoring . . . . .	95
	<b>References</b>	<b>96</b>

# List of Tables

2.1	<b>Disaggregation Accuracies on REDD</b>	39
2.2	<b>Variation of Accuracy with Depth</b>	39
2.3	<b>Disaggregation Accuracies on Pecan Street</b>	43
2.4	<b>Energy Error For Common Devices</b>	44
2.5	<b>Variation of Error with Depth</b>	44
2.6	<b>Description of Devices</b>	45
2.7	<b>Energy Error for Common Devices</b>	45
2.8	<b>TRAINING MODE DISAGGREGATION ACCURACY (MEAN OF 4 TEST HOUSES)</b>	50
2.9	<b>TESTING MODE DISAGGREGATION ACCURACY</b>	51
3.1	<b>SRC: Classification Results on REDD</b>	63
3.2	<b>SRC: Classification Results on Pecan Street</b>	63
3.3	<b>ML-SRC: Appliance-Level Evaluation on REDD</b>	68
3.4	<b>ML-SRC: Appliance-Level Evaluation on Pecan Street</b>	68
3.5	<b>MLC-RBM: Appliance-Level Evaluation on REDD</b>	70
3.6	<b>MLC-RBM: Appliance-Level Evaluation on Pecan Street</b>	70
3.7	<b>MLC-RBM: Performance Evaluation on REDD</b>	70
3.8	<b>MLC-RBM: Performance Evaluation on Pecan Street</b>	71
3.9	<b>Deep-CTL: Classification Results on REDD</b>	72
3.10	<b>Deep-CTL: Classification Results on Pecan Street</b>	72

4.1	<b>cNILM: Disaggregation Performance Evaluation (Using Precision/Recall)</b> . . . . .	87
4.2	<b>Deep-cNILM: Classification Results on REDD</b> . . . . .	88
4.3	<b>Deep-cNILM: Classification Results on Pecan Street</b> . . . . .	89
4.4	<b>Comparison of Runtimes in Seconds</b> . . . . .	90
4.5	<b>Deep-cNILM: Classification Results on Pecan Street</b> . . . . .	90
4.6	<b>Comparison of Reconstruction</b> . . . . .	90

# List of Figures

2.1	Dictionary Learning . . . . .	20
2.2	Deep Dictionary Learning . . . . .	21
2.3	Energy Disaggregation : Qualitative Look. Left – DDSC [8]; Right – Proposed Greedy Method. . . . .	40
2.4	Comparison of Disaggregation Accuracy . . . . .	49
2.5	Comparison of Normalized Error for (left to right) - AC, Refrigerator and Washer. . . . .	49
3.1	Proposed architecture for NILM using multi-label classification RBM. . . . .	65
3.2	Training reconstruction errors of MLC-RBM. . . . .	69

# Chapter 1

## Introduction

Over the past decade, numerous research studies [1, 2] about the impact of energy demands on global warming and public health have cropped up. According to the United State Energy Information Administration, U.S. commercial and residential buildings consume about 40% of energy produced in the U.S. [3]. This considerable energy demand has raised the concern about implementing better utility systems which can provide effective energy-saving and cost-cutting schemes. It has been reported in various studies like [4] that delivering significant granular information to the users about their energy consumption leads up to 15% of energy saving. Currently, the only data (other than monetary) provided to users is total energy consumed in their buildings. The actionable feedbacks, in the form of appliance-level consumption, comparison with previous energy bills, should be delivered to the consumers. It will help the consumers to take required actions for achieving better energy-savings. The requirement of a feedback based demand management system has magnified the relevance of non-intrusive load monitoring [5] in buildings.

The utilities should be equipped with systems that can process the load measurements in buildings to provide significant information to the users. This processing will also help the utilities to float schemes that can scatter the peak-hour demands. A utility should not entirely rely on consumer's endeavours to make plans favourable. Instead, it should cater automated energy management system in buildings, which requires the minimal attention of consumers and may retain their interest in demand reduction strategies, especially during peak hours of demand. Integration of NILM with the Internet of Things (IoT) can provide a platform to consumers to connect with utilities as well as with the appliances operating in their buildings. Consumers can check the state (ON/OFF) of appliances operating in their buildings and take the required action, even in their absence. Similarly, utilities can inform consumers about various schemes based on their demand-response analysis. With increasing acceptance of IoT, it would be easy to engage consumers in energy-saving and cost-cutting activities.

## **1.1 Problem Statement**

The research problem addressed in this thesis is concerned with the task of segregating the combined energy signal of a building into the energy consumption of individual appliances. The aim is to make consumers more informed about their energy consumption through budget-friendly methods with minimal intrusion.

Currently, residential and commercial buildings account for 40 % of total



energy consumption [6], and studies have estimated that 20% of this consumption could be avoided with improvement in user behavior [7]. Disaggregation presents a way in which consumption patterns of individuals can be learned by the utility company. This information would allow utility to provide feedback to the consumer, with the goal of increasing consumer awareness about energy usage. Studies have shown that this is sufficient to improve consumption patterns [8].

## 1.2 Background

The earliest NILM techniques were based on using real and reactive power measured by residential smart meters. The appliances' power consumption patterns were modelled as finite state machines [5]. These techniques were successful for disaggregating simple two state and multistate appliances, but they performed poorly in the case of time-varying appliances which do not show a marked step increase in the power. Even in recent times, there are techniques that primarily disaggregate based on jumps and drops in the power signature [9, 10].

More recent techniques, based on stochastic finite state machines (Hidden Markov Models) [11], have improved upon the prior approach. Perhaps the most modern approach is based on learning a basis for individual appliances. Sparse coding and dictionary learning based approaches like [12] fall under this category. Another study introduced the powerlet technique to learn energy signatures [13]; this combines dictionary learning with prior assumption regarding

the time.

Most of the prior load disaggregation techniques are learning-based approaches where a disaggregating model is trained using appliance-level consumption and then load segregation is carried out. So the first stage of this processing has to be appliance-level data collection which itself is an intrusive and expensive procedure as this requires installing a sensor on each device operating in the building. Some recent studies [14], [15] propose methods that can circumvent the necessity of appliance-level load measurements. They frame NILM as a multi-label classification(MLC) problem for simultaneous identification of active appliances given only the aggregated load measurements and corresponding labels. The label contains information about the state of appliances at any given time instance, i.e. which all appliances are ON or OFF. This way, the training phase becomes non-intrusive, as only labelled aggregated load is sufficient for training the model. In most of the commercial buildings, appliances are manually controlled with a specific usage pattern. In such buildings, data acquisition for an MLC task can be easily accomplished.

We discuss some of the benchmark techniques and their shortcomings in detail in the coming subsections.

### **1.2.1 Event-based Methods**

Conventional techniques of NILM [5] has typically three stages, namely; 1. Data acquisition 2. Feature extraction, and 3. Load identification. Based on

the transmission rate of sensors employed for data acquisition, NILM can be divided into two categories. One that works with high-frequency data, typically collected at a transmission frequency higher than 1 Hz whereas the other category handles low transmission frequency (less than 1 Hz).

Papers [16], [17] are example of techniques that work with high frequency data. These techniques are based on the assumption that, in a small time interval, only two or three appliances may change their state and the key idea is to detect the transition events. This assumption is correct only for the residential sector but not for commercial buildings. Moreover, devices have overlapping steady-state characteristics, so segregation of the events resulting from two different devices may be highly difficult even if they occur at different time instances.

Event detection based NILM techniques can sustain only with high-frequency data as it is easy and decisive to locate events in data which has lesser interleaving samples. However, the acquisition of high-frequency data is very expensive.

### **1.2.2 Finite State Machines**

These days most of the appliances (like light, fan, A.C., washer) have marked different states, so it is fair to model them as HMMs. The study [11] models aggregated load as an outcome of the interaction of a finite number of independent Hidden Markov processes. Firstly, the number of hidden states in which devices may be present is determined by using a transition probability matrix where the current state of each device is dependent on the previous state (Marko-

vian Property). The sequence of aggregated loads is considered as an observed variable whereas appliance-level load gives an idea about the hidden states/state transitions responsible for the given observation sequence.

While training the model, probability distributions of each observed variable, as well as state transitions, are computed. Given a trained model, the probability of observation sequence/aggregated load is maximised to estimate appliance-level load. Usually, the Viterbi Algorithm is used to find out the optimum path of hidden states that may be responsible for new observations given the trained model.

Most of the modern appliances such as printers, computers, inverters do not have marked states. They are continuously varying. In such situations, the HMM assumption fails; this, in turn, leads to poor disaggregation performance. Some other HMM-based techniques [18], [19] are also proposed, but these models have shortcomings like susceptibility to local optima and drop in accuracy with an increase in the number of appliances.

### 1.2.3 Sparse Coding

Sparse Coding is a dictionary learning technique that allows us to learn a sparse representation of a data matrix along with its reconstruction bases. The application of dictionary learning in NILM was introduced by Kolter et al. [12]. It makes the standard assumption that there is training data collected over time, where the smart meter logs only consumption from a single device only. This

can be expressed as  $X^i$  where  $i$  is the index for an appliance, the columns of  $X^i$  are the readings over a period of time. For each appliance, a dictionary is learnt i.e., Non-negative Sparse Coding objective function is given by:

$$\min_{D^i \geq 0, Z^i \geq 0} \|X^i - D^i Z^i\|_F^2 + \lambda \sum_{p,q} (Z^i)_{pq} \quad \text{s. t. } \|d_{(j)}^i\|_2 \leq 1, j = 1, 2, \dots, n \quad (1.1)$$

where  $D^i$  is the reconstruction bases and  $Z^i$  contains the activations of these bases,  $d_{(j)}^i$  indicates  $j^{th}$  column of the dictionary for  $i^{th}$  appliance and  $n$  is the total number of vectors in the bases. Since energy is a non-negative quantity, non-negativity constraint is enforced on dictionaries as well as the coefficients. After minimizing the objective over  $D^i$  and  $Z^i$  alternatively for each class  $i = 1, 2, \dots, k$ , we can disaggregate signal by solving the following optimization problem

$$\hat{Z}^{1:N} = \arg \min_{Z^{1:N} \geq 0} \underbrace{\| \bar{X} - [D^1 | \dots | D^N] \begin{bmatrix} Z^1 \\ \dots \\ Z^N \end{bmatrix} \|_F^2 + \lambda \sum_{i,p,q} \|Z_{pq}^i\|}_{F(\sum_{i=1}^N \bar{X}, D^{1:N}, Z^{1:N})}} \quad (1.2)$$

Then predicted energy usage by  $i^{th}$  device is given by

$$\hat{X}^i = D^i \hat{Z}^i \quad (1.3)$$

and the disaggregation error is given by

$$E(X^{1:N}, D^{1:N}) \equiv \sum_{i=1}^N \|X^i - D^i \hat{Z}^i\|_F^2 \quad s.t. \quad \hat{Z}^{1:N} = \arg \min_{Z^{1:N} \geq 0} F\left(\sum_{i=1}^N \bar{X}, D^{1:N}, Z^{1:N}\right) \quad (1.4)$$

One of the disadvantages of using Sparse Coding for energy disaggregation is that the bases are not learned to minimize the disaggregation error [12]. It is assumed that the bases are distinct enough to produce small disaggregation error, which is not the case always.

#### 1.2.4 Discriminative Disaggregating Sparse Coding

Kolter et al. [12] also proposed an extension of sparse coding to improve its performance. This algorithm is known as Discriminative Disaggregating Sparse Coding (DDSC). In case of DDSC, reconstruction bases (learned from sparse coding) are discriminatively optimized to produce sparse set of activations that can produce low disaggregation error.

The regularized disaggregation error objective for DDSC becomes

$$\begin{aligned} \tilde{E}_{reg}(X^{1:N}, D^{1:N}, \tilde{D}^{1:N}) &\equiv \sum_{i=1}^N \|X^i - D^i \hat{Z}^i\|_F^2 + \lambda \sum_{p,q} (\hat{Z}^i)_{pq} \\ &\text{subject to } \hat{Z}^{1:N} = \arg \min_{Z^{1:N} \geq 0} F\left(\sum_{i=1}^N X, \tilde{D}^{1:N}, Z^{1:N}\right) \quad (1.5) \end{aligned}$$

where bases,  $D^{1:N}$  and activations,  $Z^{1:N}$  are same as those trained from Sparse

Coding while  $\tilde{D}^{1:N}$  are discriminatively optimized bases such that it can produce activations  $\hat{Z}^{1:N}$  which are as close as possible to the activations that can produce minimum disaggregation error.

Activations that can produce minimum disaggregation error is given by

$$Z^{*i} = \arg \min_{Z^i \geq 0} \|\bar{X} - [\tilde{D}^1 | \dots | \tilde{D}^N] \begin{bmatrix} Z^1 \\ \dots \\ Z^N \end{bmatrix}\|_F^2 + \lambda \sum_{i,p,q} \|Z_{pq}^i\| \quad (1.6)$$

The structured perceptron algorithm is used to train the reconstruction bases so that the activations obtained from optimizing Equation (1.2) may get as close to  $Z^{*1:N}$  as possible.

For structured prediction task  $\bar{X}$  is the input,  $Z^*$  is the desired output and  $\tilde{D}$  are the model parameters which are initialized with the reconstruction bases. Now, perceptron update [20] is performed with the step size  $\alpha$ ,

$$\tilde{D} \leftarrow \tilde{D} - \alpha(\Delta_{\tilde{D}^{1:N}} F(\bar{X}, \tilde{D}^{1:N}, Z^{*1:N}) - \Delta_{\tilde{D}^{1:N}} F(\bar{X}, \tilde{D}^{1:N}, \hat{Z}^{1:N})) \quad (1.7)$$

which becomes

$$\tilde{D} \leftarrow \tilde{D} - \alpha((\bar{X} - \tilde{D}\hat{Z})\hat{Z}^T - (\bar{X} - \tilde{D}Z^*)Z^{*T}) \quad (1.8)$$

Only positive part is kept and the atoms are re-normalized to unit norm.

### 1.2.5 Neural Networks

In paper [21], Srinivasan et al. propose single hidden layer Multilayer Perceptron (MLP) to identify multiple loads using current measurements from the incoming supply. Harmonic signatures extracted from the current waveforms of an individual device are used to train the network and segregation is performed on smart meter current measurements. However, the experiments are performed on a tiny dataset and require high -frequency data. More recently in [22], Kelly et al. made adaptations in three deep neural networks to perform load segregation.

In the first architecture, LSTMs are used to learn features of each of the appliances from the training samples and then it estimates the appliance-level load given the aggregated load. The second architecture consists of a dAE, which extracts an appliance's load from an aggregated sample, by considering the consumption of other appliances as the signal's noise component. The last architecture is a standard neural network that regresses the start time, end time and average power demand for each activation of an appliance. A significant shortcoming of these techniques is that they require appliance-level load for the training of the models.

We can observe that limited research has been carried out to apply modern machine learning techniques to NILM. This is because of the fact that MLPs are data hungry. So in problems such as NILM, where availability of data is an issue, these techniques are not very successful.



### 1.2.6 Multi-Label Classification Based Approaches

Multi-label classification (MLC) task is a variant of multi-class problem. In multi-class classification, one sample is mapped to exactly one of the labels whereas, in MLC, one sample may belong to one or more labels. Hence, in MLC, each sample is mapped to a binary vector of 0's and 1's, assigning 0 or 1 to each label. Various MLC techniques find their application in medical diagnosis, bioinformatics, and text categorisation.

Since the aggregated load of a building at an instance may be an outcome of several active appliance's loads, Tabatabaei et al. [14], and Li et al. [15], framed NILM as an MLC problem. [14] compared the performance of two multi-label classifier viz Multi-Label k-nearest neighbours (ML-kNN) and Random k-Label Sets (RakEl) using time-domain and wavelet-domain features of appliances. These algorithms belong to two different group of MLC techniques namely Problem transformation techniques and Algorithm adaptation techniques [23].

Problem transformation techniques translate multi-label problem to a multi-class classification problem. The most appropriate way to do it is to train a binary classifier for each of the label in the label set,  $L \in l$  (similar as in a multi-class classification problem) using one versus all approach. Final classification can be done by taking a union of the output of each binary classifier. RakEl [24] is an example of such classifiers.

Algorithm adaptation techniques are adapted version of single-label classi-

fication techniques to perform MLC. An adapted version of k-NN [25] which is a lazy learning technique for binary classification, known as ML-kNN is an example of such techniques.

Another recent work [26] uses Multi-label Consistent Deep Dictionary Learning and Multi-label Consistent Deep Transform Learning for simultaneous detection of active appliances from smart meter data. In this work, authors are jointly learning the representations and a mapping between the labels and learned representations using Synthesis formulation as well as Transform formulation.

These methods do not directly segregate appliance-level load but first identify states of appliances and then disaggregated load is obtained by multiplying the average power consumption of appliance with the number of instances, it was identified to be in an active state.

## **1.3 Datasets**

### **1.3.1 Reference Energy Disaggregating Dataset**

It [27] is a moderate size publicly available dataset for electricity disaggregation. The dataset consists of power consumption signals from six different houses, where for each house, the whole electricity consumption as well as electricity consumptions of about twenty different devices are recorded.

The signals from each house are collected over a period of two weeks with a high frequency sampling rate of 15kHz. In the standard evaluation protocol,

the 5<sup>th</sup> house is omitted since the data from this one is insufficient.

### 1.3.2 Pecan Street Dataset

In this thesis, we worked with a subset of Dataport dataset available in non-intrusive load monitoring toolkit (NILMTK) [28], which contains 1 minute circuit level and building level electricity data from 240 houses. The data set contains per minute readings from 18 different devices: air conditioner, kitchen appliances, electric vehicle, and electric hot tub heater, electric water heating appliance, dish washer, spin dryer, freezer, furnace, microwave, oven, electric pool heater, refrigerator, sockets, electric stove, waste disposal unit, security alarm, washer dryer.

## 1.4 Research Contributions

This thesis has three main objectives: 1. To propose more accurate algorithms for NILM than state of the art; 2. To propose an algorithm that can work with compressed energy signals in order to save the bandwidth; 3. To propose the methods which make the training phase completely non-intrusive, i.e., to dodge the requirement of the sub-metered data.

Our contributions towards these objectives are as follows:

We propose Deep Sparse Coding and Analysis Co-Sparse Coding for NILM. The usual technique is to learn a dictionary for every device and use the learned dictionaries as a basis for blind source separation during disaggregation. Prior

studies in this area are shallow learning techniques, i.e., they learn a single layer of dictionary for every device. In this work, we learn multiple layers of dictionaries for each device. These multi-level dictionaries are used as a basis for source separation during disaggregation. We show that this algorithm outperforms the benchmark techniques like Factorial Hidden Markov Model and Discriminative Disaggregating Sparse Coding.

Second, we follow the multi-label classification based paradigm for NILM and determine the state(On/Off) of the appliances present in the building. For this, we propose several algorithms that adapt Sparse Representation Classifier, Restricted Boltzmann Machine and Convolutional Transform Learning to perform multi-label classification and subsequently disaggregating the appliance-level load.

Third, we propose a compressive sampling(CS) approach. The high-frequency power signal from a smart meter is encoded (by a random matrix) to very few samples making the signal suitable for WAN transmission without choking network bandwidth. CS guarantees the recovery of the high-frequency signal from the few transmitted samples under certain conditions. This work shows how to recover the signal and simultaneously disaggregate it.

## **Chapter 2**

# **Deep Sparse Coding and Analysis**

# **Co-Sparse Coding for Non-Intrusive Load Monitoring**

This chapter presents the concept of Deep Sparse Coding(DSC) and analysis co-sparse coding for NILM. DSC is a synthesis approach of modelling the devices. Instead of learning a single level of basis / dictionary we learn multiple layers – leading to the proposed paradigm of DSC. This gives the ‘depth’ in sparse coding. The concatenated multi-layered basis is used for energy disaggregation.

Currently deep learning is the de facto standard tool in machine learning research applied to computer vision, speech processing, information retrieval (and to certain problems in natural language processing). The reason, why deep learning is successful is not well understood. There is no theoretical analysis, but astounding volume of empirical result to justify its use. Motivated by its success in allied fields, we propose a new deep learning tool- ‘deep sparse cod-

ing’. As in other areas, improvement in results justifies the motivation for going ‘deep’.

The other technique i.e., Analysis Co-Sparse Coding is an analysis approach for modelling the devices. This work follows from the success of analysis dictionary learning [29] over its synthesis counterpart in image processing. The main advantage of the analysis formulation is that, it is less prone to overfitting.

This would mean that we need less training data from households. Subsequently, this would require installing sensors on fewer homes, or installing them for fewer days on the same home. In either case, this brings down the cost of sensor deployment drastically.

This chapter is organized as follows. Section 2.1 discusses the existing work. Proposed algorithms are presented in section 2.2 and section 2.3. The experimental evaluations and results are discussed in section 2.4.

## 2.1 Literature Review

### 2.1.1 Synthesis Sparse Coding

Dictionary learning is a synthesis approach. Given the data ( $X$ ) it learns a single level dictionary ( $D$ ) such that the it can regenerate / synthesize the data from the learnt coefficients ( $Z$ ). This is expressed as,

$$X = DZ \tag{2.1}$$

Equation (2.1) is a typical matrix factorization problem. In sparse coding, the objective is to have a sparse coefficients matrix  $Z$ .

K-SVD [30] is perhaps the most popular algorithm for sparse coding. It solves a problem of the form –

$$\min_{D,Z} \|X - DZ\|_F^2 \text{ s.t. } \|Z\|_0 \leq \tau \quad (2.2)$$

We have abused the notation slightly; the  $l_0$ -norm is defined on the vectorised version of the coefficient ( $Z$ ). Here  $\tau$  defines the sparsity level of the coefficients.

K-SVD is a good algorithm, but is relatively slow owing to the necessity of computing SVDs in every iteration and running a slow orthogonal matching pursuit algorithm for sparse coding. Practical applications of dictionary learning solve an unconstrained version of Equation (2.2) with an  $l_1$  -norm for promoting sparsity.

Dictionary learning has enjoyed immense popularity in the last decade. It has been the de facto tool for solving many inverse problems in signal and image processing. Machine learning researchers used supervised variants of dictionary learning for many computer vision problems.

### 2.1.2 Analysis Co-Sparse Coding

In co-sparse analysis dictionary learning [29], the signal is analysed to generate the sparse coefficients. The solution is framed such that the sparse coeffi-

icients are not obtained; rather a clean version of the data  $\hat{X}$  is obtained so that, when operated on by the analysis dictionary  $D$ , sparse coefficients are produced. Mathematically the learning is represented as,

$$\min_{D, \hat{X}} \left\| X - \hat{X} \right\|_F^2 \text{ s.t. } \left\| D\hat{X} \right\|_0 \leq \tau \quad (2.3)$$

Here  $D$  is the analysis dictionary; it is different from the synthesis dictionary of Equation (2.1). There should not be any confusion between the two, since the context / model is different.

The analysis K-SVD algorithm is not as popular as its synthesis counterpart is mainly because it has an inefficient implementation. But it enjoys nice theoretical advantage over its synthesis counterpart.

A little analysis shows that for a synthesis dictionary of size  $m \times n$ , with sparsity (number of non-zero elements in  $Z$ )  $k$ , the number of sub-spaces is  $\binom{n}{k}$  for  $k$ -dimensional subspaces. For analysis dictionary learning of size  $p \times d$ , with co-sparsity  $l$  the number of sub-spaces is  $\binom{p}{l}$  for sub-spaces of dimension  $d - l$ . If we assume equal redundancy, i.e.,  $p = n = 2d$ , and equal dimensionality of the sub-space, i.e.,  $k = d - l$ , the number of analysis sub-spaces will be  $n$  whereas the number of synthesis sub-spaces are  $k \log_2(n/k)$  (via Stirling's approximation); usually  $n \gg k \log_2(n/k)$ . For example with  $n = 700$ ,  $l = 300$  and  $k = 50$ , the number analysis subspaces are 700 whereas the number of synthesis sub-spaces are only 191. This analysis means that for an analysis and a synthesis dictionary of same dimensions, an analysis dictionary is able to



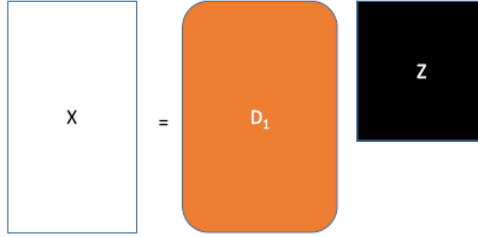


Figure 2.1: Dictionary Learning

capture significantly more variability in the data compared to its synthesis counterpart. In other words, for a fixed training set a smaller sized transform need to be learned compared to a dictionary. From the machine learning perspective, given the limited training data, learning fewer parameters for the transform has less chance of over-fitting than learning a larger number of synthesis dictionary atoms. Hence, for limited training data, as is the case with most practical problems, transform learning can be assumed to yield better generalizability (and hence better results) compared to dictionary learning. This is the motivation behind our analysis formulation.

Analysis dictionary learning has only seen a handful of applications in the past [31], [32]. But wherever they have been used (super-resolution [31], MRI reconstruction [32]), they have surpassed synthesis dictionary learning formulations. Success of such practical studies also motivates this work.

## 2.2 Proposed Formulation for Deep Sparse Coding

The popular interpretation for dictionary learning is that it learns a basis ( $D_1$ ) along with the coefficients ( $Z$ ) to represent the data ( $X$ ) (see Figure 2.1); for

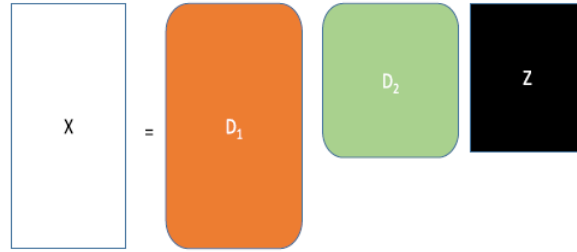


Figure 2.2: Deep Dictionary Learning

sparse coding, the representation ( $Z$ ) needs to be sparse. The columns of  $D_1$  are called ‘atoms’. Till date dictionary learning / sparse coding had been a shallow architecture. The dictionary ( $D_1$ ) is learnt such that the features ( $Z$ ) synthesize the data ( $X$ ) along with the dictionary. This is expressed as,

$$X = D_1 Z \quad (2.4)$$

We propose to extend the shallow learning into multiple layers – leading to deep sparse coding (Equation 2.2). Instead of learning one level of dictionary / basis, we learn two levels as depicted in the figure above. Mathematically, the representation at the second layer can be written as:

$$X = D_1 D_2 Z \quad (2.5)$$

It is not possible to collapse the two dictionaries  $D_1 D_2$  (Equation (2.5)) into a single level of dictionary,  $D_1$  (Equation (2.4)); the two formulations would not be equivalent. This is because Equation (2.4) is a bi-linear problem (two variables  $D_1$  and  $Z$ ) whereas Equation (2.5) is a tri-linear problem (three variables  $D_1, D_2, Z$ ); therefore the features obtained from Equation (2.4) would not be the same as those of Equation (2.5) even if the dimensions match. In Equation

(2.5) we show two levels of dictionaries; we can go deeper, to 3 and 4 layers; in that case deep dictionary learning can be expressed as (for  $N$  layers),

$$X = D_1 D_2 \dots D_N Z \quad (2.6)$$

There is no theoretical reason for finding deeper representations. However, proponents of deep matrix factorization [33], [34] argue that by finding deeper representations one can find more compact and abstract features that aids in the learning task. Usually there is a trade-off between going deeper and over-fitting. As one goes deeper, more and more parameters need to be learnt; thus the requirement for training data increases (leads to over-fitting). To prevent this one needs to find a compromise between abstraction and over-fitting. Usually this is found empirically. For most moderate size problems, a three-layer architecture is used.

There are two ways to solve Equation (2.6). The first one is a greedy approach. This is easy since the basic building blocks (shallow dictionary learning) are already available. But the limitation of this technique is that there is no feedback between the layers, i.e., the information flows from the shallower to the deeper layers but not vice versa. The second solution (the exact solution) has not been hitherto solved. In this work we solve it variable splitting followed by alternating minimization. We will discuss both the solutions in the next two sub-sections. In the exact solution, information flows across all the layers.

### 2.2.1 Greedy Solution

This is the easier of the two solutions. Here, for the first layer, we express:

$Z_1 = D_2 \dots D_N Z$ ; so that Equation (2.6) can be formulated as,

$$X = D_1 Z_1 \quad (2.7)$$

The coefficient  $Z_1$  in the first layer is not sparse, hence the learning problem can be phrased as,

$$\min_{D_1, Z_1} \|X - D_1 Z_1\|_F^2 \quad (2.8)$$

This is solved by alternating minimization.

$$Z_1 \leftarrow \min_{Z_1} \|X - D_1 Z_1\|_F^2 \quad (2.9a)$$

$$D_1 \leftarrow \min_{D_1} \|X - D_1 Z_1\|_F^2 \quad (2.9b)$$

Iterations are continued till local convergence. In the second layer, we substitute  $Z_2 = D_3 \dots D_N Z$ , leading to

$$Z_1 = D_2 Z_2 \quad (2.10)$$

As before, this is solved via alternating minimization. Substitutions are continued till the last layer. In the final layer, the formulation turns out to be,

$$Z_{N-1} = D_N Z \quad (2.11)$$

Here, the coefficient needs to be sparse. Hence the alternating minimization turns out to be the same as sparse coding (Equation 2.2).

This is an easy approach. The basic building blocks for solving this problem are well studied. There are theoretical studies on single layer dictionary learning that prove optimality of alternating minimization regarding convergence (to local minima) [35], [36]. But the problem with the greedy approach is that, information flows only in one direction – from shallow to deep; there is no feedback from latter layers to previous ones. For example one can see that  $Z_1$  (implicitly  $D_1$ ) is used for solving  $D_2$  and  $Z_2$ , but  $Z_2$  (or  $D_2$ ) does not have any influence on the solution of  $Z_1$  and  $D_1$ . This is what we mean by no feedback between the layers.

Usually in deep learning, this issue is addressed by fine tuning. However there is no scope of fine-tuning for our problem since it is an unsupervised problem – there are no targets / outputs from which one can back-propagate. Besides, dictionary learning / sparse coding is not a smooth optimization problem (not differentiable everywhere owing to the  $l_1$ -norm), hence simple gradient descent based techniques like back-propagation will not work. The exact solution is derived in the next sub-section.

### 2.2.2 Exact Solution

Our goal is to solve Equation (2.6). We have discussed the greedy approach. The exact solution is expressed as,

$$\min_{D_1, D_2 \dots D_N, Z} \|X - D_1 D_2 \dots D_N Z\|_F^2 + \lambda \|Z\|_1 \quad (2.12)$$

An elegant way to address this problem is to use the Split Bregman approach [37]; variable splitting is a standard technique in signal processing these days [38], [39]. We substitute  $Y_1 = D_2 \dots D_N Z$  and in order to enforce equality at convergence, introduce the Bregman relaxation variable,  $B_1$ . This leads to,

$$\min_{D_1, D_2, D_3, Z, Y} \|X - D_1 Y_1\|_F^2 + \mu_1 \|Y_1 - D_2 \dots D_N Z - B_1\|_F^2 + \lambda \|Z\|_1 \quad (2.13)$$

To simplify Equation (2.13) we substitute,  $Y_2 = D_3 \dots D_N Z$  and introduce another Bregman relaxation variable. This leads to,

$$\min_{D_1, D_2 \dots D_N, Z, Y_1, Y_2} \|X - D_1 Y_1\|_F^2 + \mu_1 \|Y_1 - D_2 Y_2 - B_1\|_F^2 + \mu_2 \|Y_2 - D_2 \dots D_N Z - B_2\|_F^2 + \lambda \|Z\|_1 \quad (2.14)$$

The process of substitution and introduction of Bregman variables can be continued till the last level. This leads to the following formulation,

$$\min_{D_1, D_2, \dots, D_N, Y_1, Y_2, \dots, Y_{N-1}, Z} \|X - D_1 Y_1 - B_1\|_F^2 + \mu_1 \|Y_1 - D_2 Y_2 - B_2\|_F^2 + \dots + \mu_{N-1} \|Y_{N-1} - D_N Z\|_F^2 + \lambda \|Z\|_1 \quad (2.15)$$

Although this is not exactly a separable problem, we can use the method of alternating directions to break it down to several simpler sub-problems. Showing it for  $N$  levels is cumbersome, so we do it for three levels without loss of generality.

$$P_1 : \min_{D_1} \|X - D_1 Y_1\|_F^2$$

$$P_2 : \min_{D_2} \|Y_1 - B_1 - D_2 Y_2\|_F^2$$

$$P_3 : \min_{Y_1} \|X - D_1 Y_1\|_F^2 + \mu_1 \|Y_1 - B_1 - D_2 Y_2\|_F^2$$

$$P_4 : \min_{Y_2} \mu_1 \|Y_1 - B_1 - D_2 Y_2\|_F^2 + \mu_2 \|Y_2 - B_2 - D_3 Z\|_F^2$$

$$P_5 : \min_{D_3} \|Y_2 - B_2 - D_3 Z\|_F^2$$

$$P_6 : \min_Z \mu_2 \|Y_2 - B_2 - D_3 Z\|_F^2 + \lambda \|Z\|_1$$

All the sub-problems,  $P_1 - P_5$ , are linear least squares problems having a closed form solution. Therefore, solving the sub-problems is straightforward. The last problem  $P_6$  is an  $l_1$ -minimization problem that can be solved efficiently using iterative soft thresholding [40].

In every iteration, the Bregman relaxation variable needs to be updated as follows,

$$B_1 \leftarrow Y_1 - D_2 Y_2 - B_1$$

$$B_2 \leftarrow Y_2 - D_3 Z - B_2$$

There are two stopping criteria for the Split Bregman algorithm. Iterations continue till the objective function converges (to a local minima). The other stopping criterion is a limit on the maximum number of iterations. We have kept it to be 200.

There are certain hyper-parameters that needs specification. In our case, the  $\mu$ 's refer to the coefficients at each level. Since there is no reason to give higher importance of one level over another, we use  $\mu_1 = \mu_2 = \mu_3 = 1$ . The parameter  $\lambda$  needs to be set; this parameter arises both for greedy as well as the exact solution. This was tuned by 10 fold cross-validation; the final value used in the work is  $\lambda = 0.05$ .

### 2.2.3 Energy Disaggregation

In energy disaggregation by sparse coding, a codebook is learnt for every appliance [12]. The codebook learnt in prior studies are shallow. In this work, we propose to learn deep codebooks for every appliance; instead of Equation (2.7) we will have for every appliance,

$$X^i = D_1^i D_2^i D_3^i Z \tag{2.16}$$

The codebook/dictionary for every appliance is learnt using the proposed technique (greedy or exact). Here we enforce the usual constraints – i) non-negativity of sparse coefficients; and ii) normalization of codebook.



Once the codebook for every appliance is learnt the disaggregation proceeds as before. The only difference between the previous shallow techniques and the proposed technique is that the codebook for each appliance is a cascade of codebooks / dictionaries.

$$\min_{Z_1, \dots, Z_N} \left\| X - [D^{(1)} | \dots | D^{(N)}] \begin{bmatrix} Z_1 \\ \dots \\ Z_N \end{bmatrix} \right\|_F^2 + \lambda \left\| \begin{bmatrix} Z_1 \\ \dots \\ Z_N \end{bmatrix} \right\|_1 \quad (2.17)$$

Once the loading coefficients are solved for, the energy consumed by individual appliances is calculated as before, i.e., multiplying the cascaded codebook with the corresponding coefficients.

### 2.3 Proposed Formulation for Analysis Co-Sparse Coding

The basic problem statement remains the same as in the synthesis case. There is training data available for each device ( $X_i$ ); along the rows it denotes the time period and along the columns it denotes the days. In this work, we modify from the synthesis to the analysis formulation. We propose three algorithms for analysis sparse coding with increasing levels of complexity. We start with the basic formulation.

---

**Algorithm 1:** Analysis Sparse Coding for NILM
 

---

- For every appliance  $i$  solve:  $\min_{D_i, \hat{X}_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1$
- 1: Initialize:  $\hat{X}_i = X_i, B_i = 1$  and  $D_i$  randomly
  - 2: Until convergence solve the following sub-problems in every loop
    - $P1 : \min_{D_i} \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2$
    - $P2 : \min_{\hat{X}_i} \left\| \begin{pmatrix} X_i \\ \sqrt{\mu}(Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu}D_i \end{pmatrix} \hat{X}_i \right\|_F^2$
    - $P3 : \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1$
  - 3: **Return**  $D_i$
- 

### 2.3.1 Simple Co-Sparse Coding

Just as in sparse synthesis coding, for each appliance ( $i$ ), we learn an analysis dictionary –

$$\min_{D_i, \hat{X}_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 \quad (2.18)$$

We use an unconstrained formulation; and employ  $l_1$ -norm for sparsity in place of the  $l_0$ -norm. The changes have been made to solve the problem more efficiently.

We propose a variable splitting technique to solve Equation (2.18). We introduce a proxy  $Z_i = D_i \hat{X}_i$ . With this substitution, Equation 2.18 is expressed as,

$$\min_{D_i, \hat{X}_i, Z_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 \quad s.t. \ Z_i = D_i \hat{X}_i \quad (2.19)$$

Solving the exact Lagrangian for Equation (2.19) is not necessary. It enforces exact equality between the variable and its proxy in each iteration. This is not required; we only want exact equality during convergence. Therefore, we for-

mulate the augmented Lagrangian instead.

$$\min_{D_i, \hat{X}_i, Z_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \mu \left\| Z_i - D_i \hat{X}_i \right\|_F^2 \quad (2.20)$$

The hyper-parameter  $\mu$  controls the degree of equality between the variable and the proxy. For a small value, the constraint is relaxed and for a high value, equality is enforced.

Usually a heuristic ‘heating’ technique is followed where one starts with a small value of  $\mu$  and progressive increases it after solving Equation (2.20).

The Split Bregman technique is a better alternative to such heuristic hyper-parameter heating. It has been used profusely in signal processing literature in the recent past (e.g. [32]). This technique introduces a Bregman relaxation variable ( $B_i$ ) in the constraint, leading to the following,

$$\min_{D_i, \hat{X}_i, Z_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \quad (2.21)$$

Here the relaxation variable is updated in every iteration. Therefore there is no need to tune the hyper-parameter progressively. The Bregman variable automatically updates itself to enforce convergence between the variable and its proxy; one only needs to fix the hyper-parameter  $\mu$  at a moderate value without much tuning.

The Equation (2.21) can be solved using the alternating direction method of multipliers [41]. It can be segregated into the following sub-problems –

$$\begin{aligned}
P1 &: \min_{D_i} \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \\
P2 &: \min_{\hat{X}_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \\
&\equiv \min_{\hat{X}_i} \left\| \begin{pmatrix} X_i \\ \sqrt{\mu} (Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2 \\
P3 &: \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1
\end{aligned}$$

Sub-problems  $P1$  and  $P2$  are simple least squares problems. They can be solved using closed form (Moore Penrose Pseudoinverse). Sub-problem  $P3$  also has a closed form solution via soft thresholding.

$$Z_i \leftarrow \text{signum}(D_i \hat{X}_i + B_i) \cdot \max \left( \left| D_i \hat{X}_i B_i \right| - \frac{\lambda}{2\mu}, 0 \right) \quad (2.22)$$

The final step is to update the Bregman relaxation variable.

$$B_i \leftarrow Z_i - D_i \hat{X}_i - B_i \quad (2.23)$$

This concludes the steps per iteration. We can see that all the steps have efficient closed form solutions. This makes our solution significantly less time consuming (by several orders of magnitude) compared to the Analysis-KSVD algorithm proposed in [29]. Our entire algorithm is succinctly given in Algorithm 1.

For disaggregation, we follow the standard model, i.e., the total power is the sum of the individual powers.

$$X = \sum_i X_i$$

As before, the summation is over the  $N$  appliances. The goal is to recover the individual components  $X_i$ 's given the learnt analysis dictionaries. We formulate disaggregation as,

$$\min_{\hat{X}_i} \left\| X - \sum_i \hat{X}_i \right\|_F^2 + \lambda \sum_i \left\| D_i \hat{X}_i \right\|_1 \quad (2.24)$$

Unlike the training phase, Equation (2.24) is a convex formulation. Using alternating minimization (for each component), iteration ' $k$ ' can be expressed as,

$$\min_{\hat{X}_i} \left\| X - \sum_{j \neq i} \hat{X}_j^{(k)} - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 \quad (2.25)$$

Here  $\hat{X}^{(k)}$  denotes the component corresponding to the  $j^{\text{th}}$  appliance that is not being updated in this sub-problem; they are all constants for this sub-problem.

Equation (2.24) is a typical total variation type minimization problem. However, such majorization minimization based techniques are inefficient. Today most studies employ the Split Bregman technique for solving such problems. We follow the same in this work.

As in the training phase, we substitute  $Z_i = D_i \hat{X}_i$ . After introducing the Bregman relaxation variable in the approximate equality constraint of the augmented Lagrangian formulation, we arrive at the following formulation,

$$\min_{\hat{X}_i, Z_i} \left\| X - \sum_{j \neq i} \hat{X}_j^{(k)} - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \quad (2.26)$$

Using alternating minimization, the sub-problems are:

$$\begin{aligned}
P1 : \min_{\hat{X}_i} & \left\| X - \sum_{j \neq i} \hat{X}_j^{(k)} - \hat{X}_i \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \\
\equiv \min_{\hat{X}_i} & \left\| \begin{pmatrix} X_i - \sum_{j \neq i} \hat{X}_j^{(k)} \\ \sqrt{\mu} (Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2 \\
P2 : \min_{Z_i} & \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1
\end{aligned}$$

The solution of these two subproblems are already discussed in the training phase. Both have closed form solutions. As before, the final step is to update the Bregman relaxation variables.

Note that in the testing phase, Equation (2.25) is for solving the power consumption from only a single appliance. The same need to be repeated for every appliance within one loop. The complete algorithm for disaggregation is given in Algorithm 2.

---

**Algorithm 2:** Simple Sparse Coding for NILM

---

1: Initialize:  $X_i^{(0)}$

2: Until convergence solve the following sub-problems in every loop in iteration 'k'

For every appliance  $i$  solve:  $\min_{\hat{X}_i} \left\| X - \sum_{j \neq i} \hat{X}_j^{(k)} - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1$

$$P1 : \equiv \min_{\hat{X}_i} \left\| \begin{pmatrix} X_i - \sum_{j \neq i} \hat{X}_j^{(k)} \\ \sqrt{\mu} (Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2$$

$$P2 : \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1$$

$$P3 : B \leftarrow Z_i - D_i \hat{X}_i - B_i$$

End iteration 'k'

3: **Return**  $D_i$

---

### 2.3.2 Distinctive Dictionaries

In the second formulation, we want to make the analysis dictionaries distinctive from each other, i.e., the dictionaries for each appliance should look different from others. This has not been made explicit in our first formulation.

To achieve this, we draw from literature on incoherent dictionary learning [42], [43]. Note that for two similar dictionaries ( $D_i$  and  $D_j$ ) the inner product between the one and the other  $D_i^T D_j$  will have high values along the diagonals and low values in the off diagonal elements. This property has been used in [39]–[41] to pose  $\|D^T D - I\|_F^2$  as the incoherence penalty. In this work, we want to minimize the similarities between dictionaries of different appliances. Therefore, we impose a penalty of the form  $\|D_i^T D_j - I\|_F^2$ . Adding this penalty to the training phase leads to,

$$\min_{D_i' s, \hat{X}_i' s} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_j - I \right\|_F^2 \quad (2.27)$$

Notice that, unlike the previous formulation, where the appliance-wise dictionaries were solved separately, the present formulation is coupled and hence all of them need to be solved simultaneously. Equation (2.27) can also be expressed as,

$$\min_{D_i' s, \hat{X}_i' s} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 \quad (2.28)$$

Here  $D_{i^c}$  consists of all the dictionaries except the  $i^{th}$  one stacked vertically one after the other;  $I_{N-1}$  is simply the identity matrix repeated  $N - 1$  times (where

$N$  is the number of appliances).

As before, using the same substitutions ( $Z_i = D_i \hat{X}$ ) of Equation (2.27) and introducing the relaxation variable, we recast Equation (2.28) in the Split Bregman formulation.

$$\begin{aligned} \min_{D_i's, \hat{X}_i's, Z_i's} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 \\ + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 \end{aligned} \quad (2.29)$$

The updates for  $\hat{X}_i$  and  $Z_i$  can be decoupled and hence remain the same as in the solution for Equation (2.27); both of them are known to have closed form updates. The change is in the solution for the dictionaries. For each appliance, one needs solving,

$$\min_{D_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \eta \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 \quad (2.30)$$

This is a simple least squares problem having a closed form solution.

The final step is to update the Bregman relaxation variable; it remains the same as before. Succinctly, the training algorithm is expressed in Algorithm 3.

Once the dictionaries are learnt during the training phase, there is no change in the disaggregation stage. It remains exactly the same as before.

### 2.3.3 Disaggregating Dictionaries

For disaggregation, we want the dictionaries corresponding to one particular appliance to express the data in a co-sparse fashion; the same dictionary should



---

**Algorithm 3:** Distinctive Analysis Sparse Coding
 

---

For every appliance  $i$  solve:  $\min_{D_i's, \hat{X}_i's, Z_i's} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \gamma \left\| D_{i^c}^T \hat{X}_i \right\|_F^2$

- 1: Initialize:  $\hat{X}_i = X_i, B_i = \mathbf{1}$ , and  $D_i$  randomly
- 2: Until convergence solve the following sub-problems in every loop for each  $i$ 
  - $P1 : \min_{D_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \eta \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2$
  - $P2 : \min_{\hat{X}_i} \left\| \begin{pmatrix} X_i \\ \sqrt{\mu} (Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2$
  - $P3 : \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1$
- 3: **Return**  $D_i$

---

not sparsely represent data signals from other appliances. Therefore, we impose an  $l_1$ -norm on  $D_i \hat{X}_i$  but a non-sparse (dense)  $l_2$ -norm on  $D_j \hat{X}_i$ . We impose these penalties on top of the distinctive penalty introduced in the last sub-section. Our formulation becomes,

$$\min_{D_i's, \hat{X}_i's} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 + \eta \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 + \gamma \left\| D_{i^c}^T \hat{X}_i \right\|_F^2 \quad (2.31)$$

The notation  $D_{i^c}$  has already been defined before. The term  $D_{i^c}^T \hat{X}_i$  promotes dense coefficients when dictionaries corresponding to other appliances are used.

Using the same proxy  $Z_i = D_i \hat{X}_i$ , and relaxing the equality constraint between the variable and proxy in by a relaxation variable, we arrive at the following,

$$\min_{D_i's, \hat{X}_i's, Z_i's} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \gamma \left\| D_{i^c}^T \hat{X}_i \right\|_F^2 \quad (2.32)$$

Using ADMM, Equation (2.32) can be split into the following sub-problems:

$$P1 : \min_{\hat{X}_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \gamma \left\| D_{iC}^T \hat{X}_i \right\|_F^2$$

$$P2 : \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1$$

$$P3 : \min_{D_i's} \sum_i \eta \left\| D_i^T D_{iC} - I_{N-1} \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2$$

$$\equiv \min_{D_i} \eta \left\| D_i^T D_{iC} - I_{N-1} \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2$$

Since for updating  $\hat{X}_i$ , the  $D_i$ 's are assumed to be constant, hence we are able to decouple  $P1$  into individual appliances. The update for the proxy  $Z_i$ 's are always decoupled ( $P2$ ). For updating  $D_i$ , all the other dictionaries are assumed to be fixed and hence we can decouple  $P3$  to its equivalent form. Sub-problem  $P1$  is a simple least squares problem. It can be expressed as follows,

$$\min_{\hat{X}_i} \left\| \begin{pmatrix} X_i \\ 0 \\ \sqrt{\mu}(Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\gamma} D_{iC}^T \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2 \quad (2.33)$$

It has a closed form solution in the form of Moore-Penrose pseudoinverse. The solution of sub-problem  $P2$  has already been discussed in the first sub-section. It requires only one step of soft thresholding. Sub-problem  $P3$  remain the same as Equation (2.30); it is a least squares problem having a closed form solution.

The last step in every iteration is to update the Bregman relaxation variable. We have already discussed that. This concludes the training stage. The algorithm is shown succinctly in Algorithm 4. For disaggregation, there is no change from the first formulation given in Section 2.3.1.

---

**Algorithm 4:** Distinctive Analysis Sparse Coding

---

For every appliance  $i$  solve:

$$\min_{D_i, \hat{X}_i, Z_i} \sum_i \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \|Z_i\|_1 + \eta \sum_{j \neq i} \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2 + \mu \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \gamma \left\| D_{i^c}^T \hat{X}_i \right\|_F^2$$

1: Initialize:  $\hat{X}_i = X_i$ ,  $B_i = \mathbf{1}$ , and  $D_i$  randomly

2: Until convergence solve the following sub-problems in every loop for each  $i$

$$P1 : \min_{D_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \eta \left\| D_i^T D_{i^c} - I_{N-1} \right\|_F^2$$

$$P2 : \min_{\hat{X}_i} \left\| \begin{pmatrix} X_i \\ 0 \\ \sqrt{\mu} (Z_i - B_i) \end{pmatrix} - \begin{pmatrix} I \\ \sqrt{\gamma} D_{i^c}^T \\ \sqrt{\mu} D_i \end{pmatrix} \hat{X}_i \right\|_F^2$$

$$P3 : \min_{Z_i} \left\| Z_i - D_i \hat{X}_i - B_i \right\|_F^2 + \frac{\lambda}{\mu} \|Z_i\|_1$$

3: **Return**  $D_i$

---

## 2.4 Experimental Evaluation

Simulations results are carried out on two benchmark datasets – REDD [27] and Pecan Street [28]. We show that our proposed simple extension achieves better performance than state-of-the-art shallow architectures.

Each training sample contains power consumed by a particular device in one day while each testing sample contains total power consumed in one day in particular house. Two metrics -1) Disaggregation Accuracy and 2) Energy Error are used to evaluate the performance of the models. The disaggregation accuracy is defined by [27] as follows,

$$Accuracy = 1 - \frac{\sum_t \sum_n \left| \hat{y}_t^{(i)} - y_t^{(i)} \right|}{2 \sum_t \bar{y}_t} \quad (2.34)$$

where  $t$  denotes time instant,  $i$  denotes the number of devices,  $y$  is the actual power consumption of a device,  $\hat{y}$  is the predicted power consumption of a de-

Table 2.1: Disaggregation Accuracies on REDD

Home	FHMM	SC	DDSC	MLC	PED	Proposed(Greedy)	Proposed(Exact)
1	46.6	57.17	58.11	57.42	46.0	60.76	64.26
2	50.8	65.42	68.25	62.91	49.2	71.05	74.93
3	33.3	41.06	42.40	32.17	31.7	43.50	48.26
4	52.0	60.25	73.76	62.29	50.9	76.75	79.02
6	55.7	58.06	53.93	51.91	54.5	61.71	64.19
<b>Aggregate</b>	<b>47.7</b>	<b>56.39</b>	<b>59.29</b>	<b>53.34</b>	<b>46.5</b>	<b>62.75</b>	<b>66.13</b>

Table 2.2: Variation of Accuracy with Depth

Home	Layer 1	Greedy Layer 2	Exact Layer 2	Greedy Layer 3	Exact Layer 3
1	57.17	57.11	58.42	60.76	64.26
2	65.42	62.25	68.91	71.05	74.93
3	41.06	45.40	46.17	43.50	48.26
4	60.25	67.76	69.29	76.75	79.02
6	58.06	59.93	61.93	61.71	64.19
<b>Aggregate</b>	<b>47.7</b>	<b>58.49</b>	<b>60.94</b>	<b>62.75</b>	<b>66.13</b>

vice and  $\bar{y}$  is the aggregated power consumption; the factor 2 in the denominator is to discount the fact that the absolute value will “double count” errors.

The Energy Error is given by the sum of the differences between the estimated energy consumption and actual energy consumption of appliance  $n$  in each time instant  $t$ , normalised by the appliance’s total energy consumption.

$$\text{Energy Error} = \frac{\sum_t |y_t^{(n)} - \hat{y}_t^{(n)}|}{\sum_t y_t^{(n)}} \quad (2.35)$$

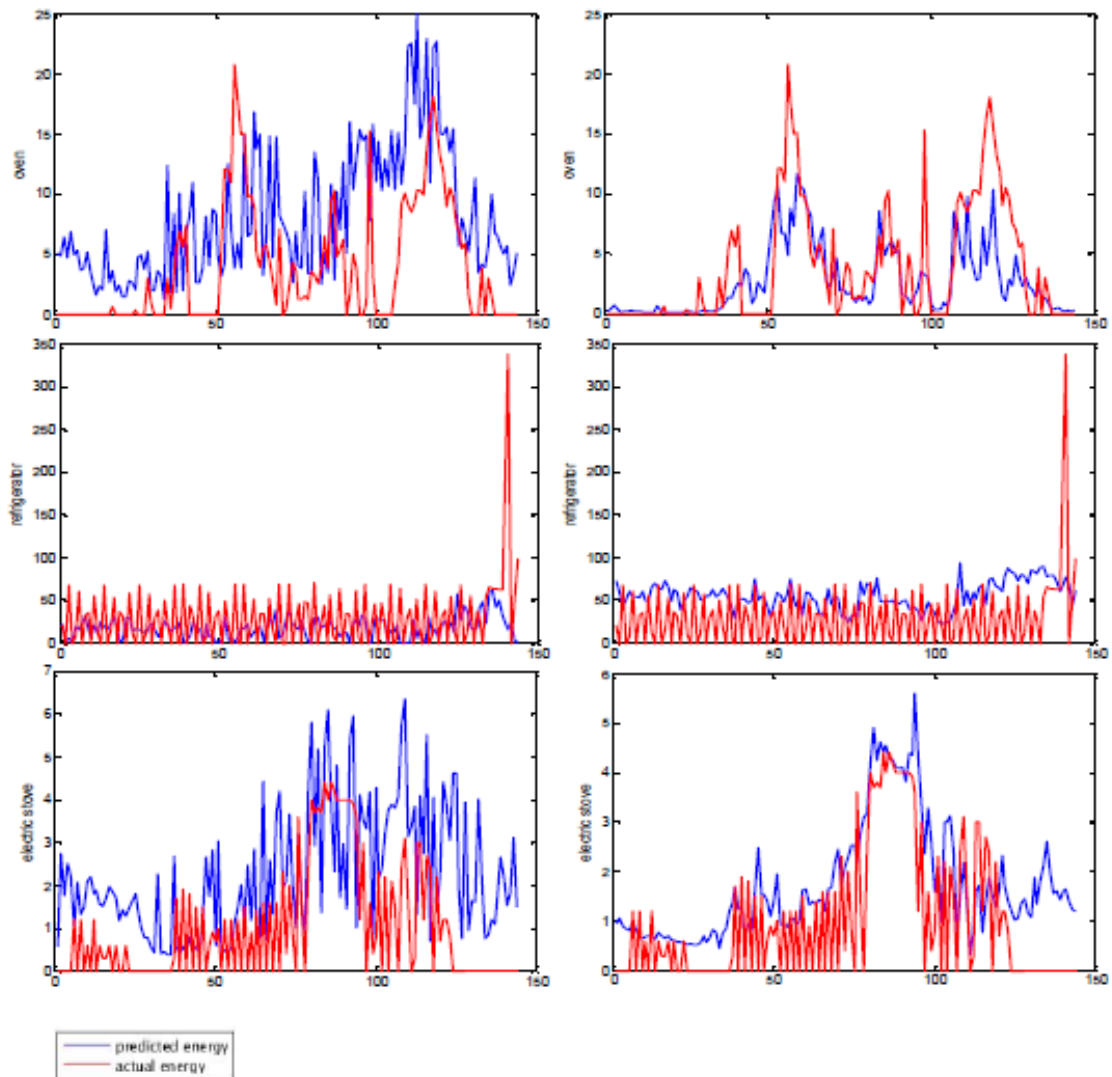


Figure 2.3: Energy Disaggregation : Qualitative Look. Left – DDSC [8]; Right – Proposed Greedy Method.

## 2.4.1 Results with Deep Sparse Coding

### 2.4.1.1 Results on REDD

To prepare training and testing data, aggregated and sub-metered data are averaged over a time period of 10 minutes.

We compare the performance of the proposed method with the Factorial HMM (FHMM) based technique [19], Powerlet based Energy Disaggregation

(PED) [13], sparse coding (SC), Discriminative Disaggregating Sparse Coding (DDSC) [12] and multi-label classification (MLC) [44]. As outlined by [12] – there are two protocols for evaluation. In the first one (called ‘training’), a portion of the data from every household is used as training samples and rest (from those households) is used for prediction; this is the easier of the two protocols. In the second mode, the data from four households are used for training and the remaining one is used for prediction (called ‘testing’); this is a more challenging problem. In this work, we carry out experiments on the more challenging problem, i.e., testing protocol.

The results are shown in Table 2.1. The SC and DDSC yields the best results for 144 atoms. For our method (both greedy and exact) the number of atoms are 144-100-80 in three layers. The table shows that our method is considerably superior compared to other benchmark disaggregation techniques.

The results are as expected. Results from discriminative sparse coding are slightly better than shallow sparse coding, but are worse compared to the proposed one. The improvement from our greedy technique is decent, but it is not the best. The results obtained from the proposed exact solution yields the best results.

In Table 2.2, the intermediate results for Layer 2 are shown. Layer 1 is the same for both the exact and greedy solution; it is the same as sparse coding. However results for layer 2 and 3 are different. One can see that from Layer 1 to 2, there is a significant improvement. But from Layer 2 to 3, the improvement

is nominal for the greedy solution. The exact solution continues to improve in the Layer 3. However, going beyond Layer 3 did not improve the results either for the greedy or the exact solution.

#### **2.4.1.2 Results on Pecan Street**

The number of atoms for different techniques remain the same as before. The results are shown in Table 2.3. The conclusion remains the same as before. Our method outperforms other techniques by a wide margin. The interesting observation here is that by deep sparse coding, we are able to get significantly larger improvement on homes where the disaggregation accuracy was previously lower, e.g., 6-8, 15, 29 etc.

For the Pecan Street dataset, we also study the variation of performance with respect to different electrical appliances. The metric used here is Energy Error. The results are shown in Table 2.4. The results show that our proposed method yields the best disaggregation in terms of normalised error for every device. FHMM and PED yields significantly worse results. Sparse coding and Discriminative Disaggregating Sparse Coding yield reasonably good results but are worse than the proposed Deep Sparse Coding.

We carried out an analysis similar to the previous subsection. In Table 2.5, variation of with dictionary-depth for major appliances is shown out. The conclusions derived remain similar. However, for this dataset, we find that even for the greedy method, there is significant drop in error from Layer 2 to 3.

Table 2.3: Disaggregation Accuracies on Pecan Street

Home	FHMM	SC	DDSC	MLC	PED	DSC (Greedy)	DSC (Exact)
1	75.55	89.43	90.53	87.94	75.96	92.96	94.09
2	42.99	65.34	66.90	63.83	43.57	74.94	79.20
3	64.13	81.50	82.02	80.22	66.21	83.64	87.82
4	51.56	61.79	71.19	60.31	52.75	74.70	79.62
5	52.20	53.49	62.14	52.00	52.69	62.50	70.05
6	10.00	54.62	54.68	53.13	13.92	52.92	60.36
7	53.75	49.03	54.61	47.60	55.06	60.44	67.84
8	32.94	51.91	52.85	50.44	33.94	60.66	66.92
9	75.50	74.27	75.35	72.88	75.06	77.40	80.40
10	46.26	56.28	63.34	54.79	48.38	67.25	71.06
11	33.05	53.59	59.30	52.07	33.69	67.37	72.30
12	44.12	65.79	69.20	64.29	45.97	71.75	75.21
13	50.25	62.97	69.63	61.49	51.11	74.80	77.34
14	70.79	82.79	84.67	81.30	72.52	87.30	90.86
15	50.93	60.73	61.21	53.22	50.62	61.98	69.51
16	74.45	85.51	86.84	84.00	75.82	88.78	90.11
17	90.15	84.94	85.64	83.45	89.91	81.12	83.40
18	57.93	75.28	75.86	73.80	58.90	77.68	81.26
19	45.74	55.67	58.93	50.20	47.00	61.90	67.89
20	48.06	59.40	64.73	57.95	48.81	69.23	74.37
21	57.87	56.58	58.67	55.09	57.03	60.73	66.80
22	35.67	50.70	52.11	49.19	38.60	48.14	56.76
23	68.75	81.30	84.28	79.81	71.26	87.69	90.09
24	62.43	75.14	78.73	73.63	65.99	85.85	89.28
25	39.44	49.76	50.20	45.16	37.59	51.89	58.23
26	31.94	49.97	51.49	48.50	32.60	53.06	59.31
27	42.68	45.40	50.54	43.90	43.11	55.50	60.75
28	68.07	77.39	78.31	75.85	69.07	79.63	84.08
29	31.00	55.65	55.65	54.14	31.00	57.11	66.02
30	35.75	53.09	55.68	52.16	38.85	55.18	63.96
31	38.81	52.09	52.92	50.19	40.03	51.44	59.82
32	47.24	63.95	67.30	61.09	59.92	71.79	75.60
33	71.00	66.88	68.69	62.91	67.06	67.25	69.22
34	31.37	48.47	50.37	45.16	33.92	49.74	58.31
35	45.36	48.95	51.10	52.60	45.90	58.74	63.50
36	26.89	44.87	49.95	50.76	30.13	52.02	58.34
37	30.73	50.68	54.51	50.11	38.71	59.42	64.31
38	38.28	60.04	61.92	55.36	41.09	62.85	65.55
39	63.95	73.79	76.91	70.26	64.06	83.15	85.82
40	47.32	52.86	53.25	53.02	50.09	51.11	61.79
41	47.51	46.19	50.76	44.92	55.03	53.10	62.06
42	51.10	61.91	65.63	60.72	51.85	68.97	72.56
43	60.70	72.52	77.94	69.26	61.37	84.83	87.24
44	28.41	55.35	56.89	50.93	29.18	58.90	65.32
45	56.53	78.47	81.79	78.61	58.51	84.66	87.09
46	35.16	49.17	54.55	45.55	39.06	61.89	69.74
47	41.75	71.46	73.67	69.93	49.38	72.67	76.77
<b>Aggregate</b>	<b>49.07</b>	<b>62.06</b>	<b>64.96</b>	<b>60.29</b>	<b>50.90</b>	<b>67.58</b>	<b>72.72</b>



Table 2.4: Energy Error For Common Devices

Appliance	FHMM	SC	DDSC	MLC	PED	DSC (Greedy)	DSC (Exact)
AC	3.16	0.90	0.70	0.81	2.52	0.89	0.80
Dryer	51.47	16.57	2.04	13.24	35.69	1.11	1.02
Dishwasher	6.48	4.23	1.25	3.19	6.08	0.66	0.62
Microwave	4.96	4.55	0.84	2.80	4.3	0.76	0.70
Furnace	0.89	0.79	0.63	0.66	0.93	0.58	0.55
Fridge	2722.8	916.53	516.3	1001.3	986.30	490.56	401.78
Washer	21.80	8.75	0.93	5.59	19.62	0.59	0.55

Table 2.5: Variation of Error with Depth

Appliance	Layer 1	Greedy Layer 2	Exact Layer 2	Greedy Layer 3	Exact Layer 3
Ac	0.90	0.90	0.84	0.89	0.80
Dryer	16.57	12.04	10.24	1.11	1.02
Dishwasher	4.23	2.25	1.19	0.66	0.62
Microwave	4.55	2.84	2.80	0.76	0.70
Furnace	0.79	0.66	0.61	0.58	0.55
Fridge	916.53	516.3	490.3	490.56	401.78
Washer	8.75	2.93	2.59	0.59	0.55

To visually show the disaggregation results for the Pecan Street dataset, some samples are shown in the Figure 2.3. The red plot shows the actual energy consumed and the blue plot the predicted energy. One can see that even with our proposed greedy method, the estimated and the actual values are close, while results from [8] are considerably off.

#### 2.4.1.3 Results on an Indian Dataset

There is hardly any dataset collected outside the developed countries. This is one of early endeavours in a developing nation (India) to collect an NILM datasets [45]. The data was collected in a three storey building in Delhi, India, spanning 73 days from May-August 2013. The brief description of the dataset is given in Table 2.6. For major appliances, electricity consumption was moni-

Table 2.6: Description of Devices

Appliance Name	Number	Power Ratings	Brief Description of Appliances
Air Conditioner	2	1800	Appliances Type: Window, Cooling Capacity: 5050W, Power Consumption(W):1778
Laptop	1	90	Not available
Television	1	50	Samsung Television Series 3, BN68025578
Water Filter	1	40	AMY JP Appliances Water Purifier
Refrigerator	1	160	LG Refrigerator, Electricity Consumption: 260 units/year, Volume:230 litres
Washer	1	NA	LG 6.5kg Fabri Soak Washing Machine WP-9515

Table 2.7: Energy Error for Common Devices

Appliance	DDSC	DSC(Greedy)	DSC(Exact Layer)
Air conditioner	0.73	0.46	0.30
Laptop	10.46	4.31	3.20
Refrigerator	2.17	0.59	0.27
Television	7.10	2.86	1.96
Water Filter	83.28	26.00	19.78
Washer	5.97	3.86	2.08

tored at three levels:

**Meter level:** Modbus-serial enabled Schneider Electric EM64001 meter was used to instrument the main power supply. The collected data includes voltage, current, frequency, phase and power at 1 Hz.

**Circuit level:** Split-core current transformers (CT), clamped to individual mini-circuit breakers, are used for monitoring circuit level current. Since no commercial solution was easily available in India for panel level monitoring, a custom built solution was used involving low cost microcontroller and Single Board Computer (SBC) platform. A total of 8 CTs were used to monitor

different MCB circuits in the home.

**Appliance level:** Since no good commercial options were available for plug level monitors, we worked with our collaborators and used their in-house developed jPlug (a variant of nplug [46]) for monitoring individual appliance level power consumption. Ten jPlugs were used to monitor different plugload based appliances across the home. It measured multiple parameters including voltage, current, phase and frequency.

Additionally, Current Cost (CC) based CT is used to measure the power consumption for electric motor (used to pump water), which is not a plug-load, but has a significant power consumption.

Different computing platforms - microcontrollers, SBCs and desktops are used for data collection. Five raspberry pi's (RPi) and one Ionics Stratus plug computer (as SBC) and a 2 GHz Desktop PC running Linux (main local server) was used. One RPi, connected to EM6400 using RS485-USB converter, collected meter data using a custom program based on pyModbus5 and communicated it to the desktop server. USB output (XML formatted) from CC is collected on another RPi and is communicated to the desktop server.

For disaggregation we used the readings at the meter level and at the appliance level; the data used in this work is aggregated and averaged to 10 minute resolution. The same three tier architecture has been used for this dataset. The disaggregation accuracy using FHMM [19] is 53.16%, using sparse coding [12] is 71.02%, using discriminative sparse coding [12] is 73.20% and us-

ing PED [13] is 57.28%. These conclude the baseline techniques. Our proposed greedy deep sparse coding yields an accuracy of 76.93% and exact deep sparse coding yields 78.04%. The improvement we get by going deep instead of using insights into the appliance's behaviour [12, 13] in a shallow technique yields significant improvement.

The disaggregation performance (in terms of error) for different devices are shown in the Table 2.7. We have shown the results for Discriminative sparse coding (which is the best among existing techniques) and the proposed deep techniques.

We compare the performance of the proposed method with two baseline techniques - FHMM, and DDSC. The rest are benchmark methods – PED , MLC , and DSC [47]. FHMM is a standardized technique and the parameters are known from the non intrusive load monitoring toolkit [28]. For the remaining, the parameter values have been obtained from the corresponding studies.

For our proposed technique, the value of the sparsity inducing parameter  $\lambda$  has always been kept at 0.1 (for all algorithms). For the incoherence term, the parameter  $\eta$  has been set to 0.2 and for the disaggregating term the value of  $\gamma$  has been fixed at 0.05. These values were obtained by cross validation on the training data using the greedy  $L$ -curve technique.

It is greedy in the sense, that the value of the common sparsity parameter is obtained from the first technique. It is kept fixed in the second technique to find out  $\eta$ ; the values of  $\lambda$  and  $\eta$  have been fixed for the third formulation for fixing

$\gamma$ . Our algorithm is not sensitive to the value of the hyperparameter for a wide range of values (between 0.01 and 0.95) – this is expected since the Bregman relaxation variable adjusts automatically. The number of atoms we have used for each device is 3.

#### 2.4.1.4 Results on REDD

As outlined by [27] – there are two protocols for evaluation. In the first one (called ‘training’), a portion of the data from every household is used as training samples and rest (from those households) is used for prediction. Usually 80 % of the data (sequentially) is used for training and the remaining for testing. In the second mode, the data from four households are used for training and the remaining one is used for prediction (called ‘testing’). The usual protocol for the testing mode is to use 4 houses for training the and 5<sup>th</sup> house for testing.

In this chapter, we have argued that the motivation for using analysis dictionary learning is its generalization ability, one requires lesser training data. Therefore, we propose more challenging protocols for testing and training modes. For the testing mode, we will use only one of the houses for training and the remaining four for testing. In the training mode we use 20% of the data for (for each house) training and the remainder for testing. The split into training and testing set has been done randomly and 100 such splits have been made. We report the mean from all the splits. In the following Table 2.8 we show results for training mode. The testing mode disaggregation accuracy is shown in Table 2.9. For both the tables ‘Simple’ is the technique proposed in Subsection 2.3.1;

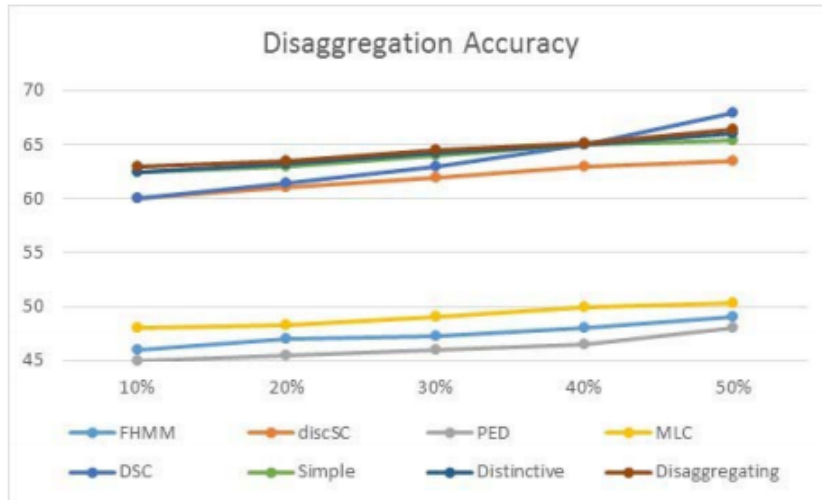


Figure 2.4: Comparison of Disaggregation Accuracy

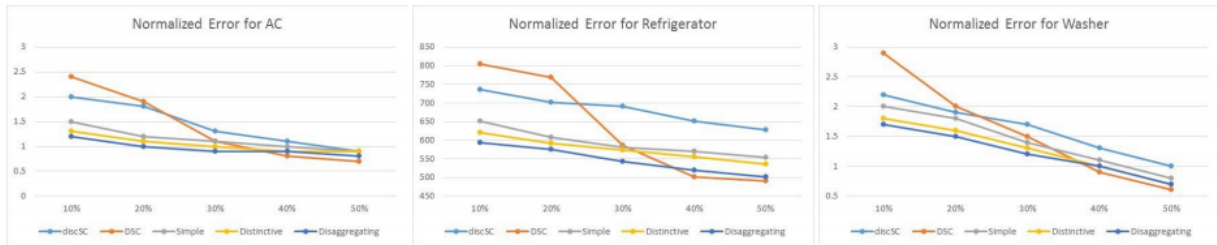


Figure 2.5: Comparison of Normalized Error for (left to right) - AC, Refrigerator and Washer.

‘Distinctive’ is the technique proposed in Subsection 2.3.2; and ‘Disaggregating’ is the technique proposed in Subsection 2.3.3.

### 2.4.2 Results with Analysis Co-Sparse Coding

The results conclusively show that our proposed methods are significantly better than others. Our baseline ‘Simple’ technique yields more than 5% improvement over the next best (PED) for the training mode and DSC for the testing. It is interesting to note that, a benchmark deep learning algorithm like DSC performs the worst. This is because, deep learning is data hungry; in limited data settings such as the training, it overfits and performs significantly worse compared to

Table 2.8: TRAINING MODE DISAGGREGATION ACCURACY (MEAN OF 4 TEST HOUSES)

Home	FHMM	DDSC	PED	MLC	DSC	Simple	Distinctive	Disaggregating
1	53.6	52.2	54.1	56.4	46.0	60.2	60.7	62.0
2	57.8	60.4	64.3	60.9	49.2	70.0	70.0	72.0
3	41.3	40.0	40.4	30.2	31.7	42.1	42.9	46.5
4	58.0	56.3	68.7	60.3	50.9	75.3	76.2	76.8
6	62.7	54.1	54.9	50.9	54.5	60.4	61.1	62.7
<b>Aggregate</b>	<b>54.7</b>	<b>52.6</b>	<b>56.5</b>	<b>51.7</b>	<b>46.5</b>	<b>61.6</b>	<b>62.2</b>	<b>64.0</b>

other shallow techniques.

However in the testing mode, since the data from all houses are aggregated, it performs better. We see that between the three different proposals of ours, there is only light difference. The ‘Simple’ method yields good results. It is slightly improved with the ‘Distinctive’ penalty; the results improve further with the additional ‘Disaggregating’ penalty. The overall improvement we achieve over the existing techniques is 7.5%.

For the Training mode, we carried out statistical t-tests between the methods in order to verify if they are significantly different from each other. At 99% confidence interval, we found that our ‘simple’ and the ‘distinctive’ techniques were statistically similar but our final formulation – the ‘disaggregating’ technique was different (better). All of our proposed techniques were significantly better than the benchmark techniques.

#### 2.4.2.1 Results on Pecan Street

In this work, we make the evaluation more challenging. We use 10% to 50% of the houses for training and the remaining for testing. The splitting into training

Table 2.9: TESTING MODE DISAGGREGATION ACCURACY

Home	FHMM	DDSC	PED	MLC	DSC	Simple	Distinctive	Disaggregating
1	46.6	46.0	44.2	43.8	50.2	55.0	55.7	58.0
2	50.8	49.2	48.7	48.5	53.4	65.1	65.2	66.8
3	33.3	31.7	30.1	31.0	38.9	37.2	37.6	40.5
4	52.0	50.9	46.3	48.2	56.8	70.5	70.9	71.9
6	55.7	54.5	50.4	51.6	59.0	55.2	55.2	57.1
<b>Aggregate</b>	<b>47.7</b>	<b>46.5</b>	<b>43.9</b>	<b>44.6</b>	<b>51.7</b>	<b>56.6</b>	<b>56.9</b>	<b>58.9</b>

and testing sets is done randomly and 100 such splits have been used in the experiments. What we report are the average of the 100 splits.

To prepare training and testing data, aggregated and submetered data are averaged over a time period of 10 minutes. This is the usual protocol to carry out experiments on the Pecan street dataset. Each training sample contains power consumed by a particular device in one day while each testing sample contains total power consumed in one day in particular house.

For the proposed techniques, number of atoms for different appliances remain the same as before (i.e., three per appliance). The parametric values also remain the same as in REDD; we did not tune it any further. The configuration of the techniques compared against are also obtained from the non-intrusive load monitoring toolkit as before. The parameter settings for the benchmark methods are from the corresponding papers.

It is not possible to give the house-wise results like REDD. Therefore we show the results through two sets of graphs. The graph in Figure (2.4) shows the overall accuracy of each technique for a given training volume. We clearly see two distinct classes of techniques. The PED, MLC and FHMM are the



bottom performing techniques; DDSC, and the proposed ones are better. Of these, DDSC is the worst. DSC performs worse than analysis co-sparse coding when the volume of training data is low, but with increase the results continue to improve and eventually surpasses ours.

The proposed methods yield the same level of accuracy with only 10% training data as compared to benchmark techniques utilizing 50% training data. This means that, given the scenario, we only need to instrument 10% of the homes as compared to 50% (required by existing methods); this is a drastic reduction in instrumentation and sensing cost.

We refer to this result (five fold reduction in the need for sensing) in the introduction while giving the example. The second set of graphs show the normalized error (a common metric) for common high power consuming appliances from different techniques. This is shown in Figure 2.5. For this set of graphs we only show results for best performing methods – DSC and DDSC; this is because the results from other techniques are so poor that the errors are larger by an order of magnitude making visual comparison meaningless. The results show that, for smaller training volume our method performs the best. As the volume of training data increases deep sparse coding tends to perform better. This is expected. Deep learning overfits with small training data and hence perform poorly.

The conclusions drawn before hold in this set of plots as well. Our proposed method performs at par with the benchmark methods with far fewer training

data; in practice this leads to far fewer instrumented homes, thus reducing the costs of sensors.

## 2.5 Discussion

Sparse coding based techniques have been shown to yield excellent disaggregation results. However, all prior sparse coding techniques are shallow, i.e., single layer of dictionary is learnt for each device. For the first time in this work we propose the concept of learning deeper levels of dictionaries; we call this – deep sparse coding. Simulations results on two benchmark datasets and experimental results on a real dataset show that our proposed method is always better than the state-of-the-art methods in energy disaggregation.

The shortcoming of our work (and all other studies based on sparse coding / dictionary learning) is that, it cannot be used for real-time disaggregation. If such be the need, HMM based techniques [48] would be more suitable. The other shortcoming, is that by going deeper, we require learning more parameters. Therefore when training data is limited, we will overfit and suffer degradation in testing performance. In the future, we would like to adopt the unsupervised pre-training followed by fine-tuning paradigm used in deep learning to ameliorate this issue.

Prior studies [12], [13] have shown that better results can be obtained (for shallow techniques) when further assumptions regarding the device are made. In future we would like to incorporate it into our deep learning framework and

hope to improve the results even further.

One of the general problems in deep learning is the unavailability of supervised data. We have shown that the deep learning framework proposed in this chapter works for databases of different sizes. But what about extreme situations where the data is highly parsimonious? In deep learning this is addressed by the paradigm of unsupervised pre-training followed by supervised fine-tuning [49]. In the future, it remains to be seen if similar techniques can be adopted for deep sparse coding based disaggregation as well.

Another new technique for energy disaggregation has been proposed in this chapter. It is based on analysis co-sparse coding. Results on benchmark databases show that the proposed technique performs better than others when the volume of training data is small. When the volume of training data is large, recently proposed method of deep sparse coding performs better.

In a practical scenario it would mean that our method will require far fewer number of instrumented houses, or far fewer days of instrumentation in each house for reaching the same level of accuracy as the benchmark techniques today. This means that using our method one can drastically reduce the sensing cost without losing on disaggregation accuracy.

## Chapter 3

# Non-Intrusive Load Monitoring via Multi-Label Classification

In this chapter, we follow a non-traditional approach of framing NILM as a multi-label classification problem [14]. As part of this work we have adapted two existing classification methods (sparse representation based classifier restricted boltzmann machine), and proposed a novel algorithm (based on convolutional transform learning) to perform multi-label classification.

Traditional NILM is not fully non-intrusive; the data collection for the training stage is highly intrusive requiring installation of sensors at the plug level to record the consumption of individual appliances over months. This training data is used to train a model, which is then used in the operational/testing stage to disaggregate the load; the operational stage is non-intrusive.

Owing to the high cost of data collection, financial and privacy-wise, large scale roll-out of NILM as a service has not been achieved. There is a need to make

the process completely non-intrusive (at least as far as sensing is concerned). The recent multi-label classification based framework for NILM is showing promise [14] in this direction. In this approach, the actual power consumption of the appliances is not required, only the ON/OFF state of the device needs to be recorded. This can be done by recording logs from individual households for residential buildings and building managers for commercial ones. Such an approach reduces both instrumentation costs and mitigates privacy concerns in one go.

In this chapter, a brief literature review is described in Section 3.1 followed by proposed methodology in Section 3.2. Then Section 3.3 presents the experiments and results.

### **3.1 Literature Review**

Over time, various approaches have been proposed to address NILM ranging from combinatorial optimization [5] and stochastic finite state machines [48] to modern deep learning based techniques [22]. A slightly dated review on this topic is available in [50], [51].

In the multi-label classification based approach the states of the appliances are the class label. The recorded smart meter readings serve as the input sample. Since multiple appliances can be ON at the same time, it turns out to be a multi-label classification problem. In [14] a thorough comparison of traditional multi-label classification algorithms like multi-label K nearest neighbour

(MLKNN) and random  $K$  label-sets (RaKEL) for NILM have been performed. More modern approaches are based on multi-label deep learning [26] and multi-label graph learning [52] for NILM; these form the state-of-the-art in this area.

In this work we propose a new approach to multi-label classification based on the sparse representation based classification (SRC) approach [53]. The technique was originally developed for computer vision, but has been widely used in various domains since then; the paper has 8000+ citations. The main advantage of SRC over other approaches is its ability to infer from very few samples. This is a critical criterion for NILM – the smaller the training data required the better it is.

Compare two scenarios for a domestic household – training data logged for 7-8 months versus data logged for one month. In the first one, it is likely that the household will refuse to participate for two main reasons:

1. logging the data is tedious
2. household cannot go for vacation in this duration

These issues are not going to arise for the second scenario where they have to log the data for just a month. This is the reason, we are emphasizing on a mechanism that can infer from small volume of training data.

Originally SRC was developed for single label classification, i.e., for problems where the input samples corresponded to only one class label. Here, we show how SRC can be easily extended to handle multi-label classification problems.

## 3.2 Proposed Formulations

### 3.2.1 Multi-Label Sparse Representation-Based Classification

SRC assumes that the test sample ( $v_k$ ) can be represented as a linear combination of training samples from the correct  $k^{th}$  class. This is represented as follows,

$$v_{test} = \alpha_{k,1}v_{k,1} + \alpha_{k,2}v_{k,2} + \dots + \alpha_{k,n_k}v_{k,n_k} + \epsilon_k = V_k\alpha_k + \epsilon_k \quad (3.1)$$

Here  $v_{k,i}$  represents training samples for the  $k^{th}$  class and  $\alpha_{k,i}$  the corresponding linear weights;  $V_k$  is formed by stacking the  $v_{k,i}$ 's as columns and  $\alpha_k$  is formed by  $\alpha_{k,i}$ 's stacked as a vector. The error  $\epsilon_k$  is assumed to be Normally distributed.

However, the correct class is not known, therefore a better way to represent the SRC model is to express the test sample as linear combination of all training samples where the weights corresponding to samples of incorrect class will be zero. This can be expressed as follows,

$$v_{test} = \sum_k V_k\alpha_k + \epsilon = V\alpha + \epsilon \quad (3.2)$$

Here  $V$  represents all the training samples stacked as columns and  $\alpha$  is formed by concatenating the  $\alpha_k$ 's vertically. The error  $\epsilon$  is Normally distributed.

According to the *SRC* assumption [53], most of the coefficients in  $\alpha$  will be zeroes. Therefore Equation (3.2) is a sparse recovery problem. One can use  $l_p$ -norm minimization ( $0 < p = 1$ ) or any greedy algorithm for solving  $\alpha$ .

Once the sparse  $\alpha$  is obtained, the task is to assign  $v_{test}$  to the correct class. In SRC this is done by computing the distance between  $v_{test}$  and the class representation defined by  $V_k\alpha_k$ . Usually a simple Euclidean distance is computed:  $d_k = \|v_{test} - V_k\alpha_k\|_2$ . It is expected that for the correct class this distance will be the smallest; therefore it is sensible to assign  $v_{test}$  to the class having the minimum  $d_k$ .

This concludes the single label SRC technique. In multilabel SRC the input test sample  $v_{test}$  may belong to multiple classes. Therefore instead of assigning the test sample to only one class by looking at the minimum  $v_{test}$ , we will consider other classes that have small  $d_k$ 's. For multi-label classification we can consider all classes within the range  $\tau \times \min(d_k)$  to be active classes for  $v_{test}$ ; in this work we have used  $\tau = 2$ . The algorithm is expressed succinctly.

### 3.2.1.1 ML-SRC Algorithm

1. Solve the optimization problem expressed in Equation (3.2).
2. For each class  $k$  compute class-wise distance:  $d_k = \|v_{test} - V_k\alpha_k\|_F^2$
3. Assign test sample to all classes whose distance is than  $2 \times \min(d_k)$ .

### 3.2.2 Multi-Label Restricted Boltzmann Machine

We propose a new approach to multi-label classification based on the Restricted Boltzmann Machine (RBM) [54]. RBMs have never been used for multi-label classification so far. It is a classic example of algorithm adaptation for multi-



label classification.

RBM [55] have been effective in learning high-level features and capturing high-order correlations of the observed variables. A typical RBM has a hidden unit in which nodes are conditionally independent given the visible state. RBMs have good reconstruction accuracy which can be leveraged to generate individual load information in latent space. We propose that generative property of RBMs combined with multi-label supervision can be used to perform NILM via state detection of appliances.

Restricted Boltzmann Machines [55] are one type of undirected graphical models that use hidden variables to model high-order and non-linear regularities of the data. A typical RBM is a two-layer bipartite graph with two types of units, the visible units  $x$  and hidden units  $h$ . An RBM represents probability distributions over the random variables under an energy-based model. The energy model of an RBM is given by  $E(x, h) = -x^T W h - b^T x - c^T h$ , where  $W$  is the weight to be learned. The joint probability distribution over  $(x, h)$  is expressed as  $P(x, h) = \frac{1}{Z} \exp(-E(x, h))$ , where  $Z$  is the normalization factor.

Learning RBMs is a difficult task due to the tractability involved in computing normalization factor  $Z$ . Several learning algorithms have been proposed [56–58] to solve the problem above. Contrastive Divergence (CD) method proposed by Hinton et al. [56] is an efficient method and is widely used to learn RBMs.

The generative property of RBM makes it useful for learning latent space

representation of data where we don't have information about how data is generated. RBMs have been used for dimensionality reduction [59], collaborative filtering [60], anomaly detection [61] and unsupervised feature learning [62]. The classification RBM has been used for various classification tasks in [54, 63] and label consistent collaborative filtering [64].

The joint probability distribution of the proposed multi-label classification RBM model shown in Figure 3.1 is given by,

$$p(y, x, h) \propto e^{-E(y,x,h)} \quad (3.3)$$

where  $y$  is the label unit. We define the new energy function as follows:

$$E(y, x, h) = -h^T W x - a^T x - b^T h - c^T y - h^T U y \quad (3.4)$$

with parameters  $\Theta = (W, a, b, c, U)$ . The model is illustrated in figure 3.1. We find the values of visible and hidden units using (3.5), (3.6) and (3.7) respectively.

$$p(h_j = 1|x, y) = \sigma(b_j + U_{jl} + \sum_k W_{jk} x_k) \quad (3.5)$$

$$p(x_i|h) = \sigma(a_i + \sum_j W_{ji} h_j) \quad (3.6)$$

$$p(y_l = 1|h) = \frac{\exp(c_l + \sum_j U_{jl}h_j)}{\sum_{l=1}^y \exp(c_l + \sum_j U_{jl}h_j)} \quad (3.7)$$

where  $\sigma$  is the logistic sigmoid and  $l$  is the class label out of  $C$  classes. These formulations capture the predictive information about the input vector as well as the target class.

Network parameter  $\Theta$  is learned using CD [56] algorithm,

$$\begin{aligned} \Delta W_{ij} &= \eta \frac{\delta \log p(x, y)}{\delta W_{ij}} \\ &= \eta (\langle x_i y_i h \rangle_{data} - \langle x_i y_i h \rangle_{model}) \end{aligned} \quad (3.8)$$

where  $\eta$  is the learning rate.

For multi-label classification RBM, the above formulation changes as now we have multi-label information present for each sample. The conditional distribution of  $y$  given  $h$  becomes:

$$p(y_l = 1|h) = \sigma(c_l + \sum_i U_{li}h_i) \quad (3.9)$$

This formulation is not tractable since  $y$  now has  $2^C$  possible values. Therefore for inference we use mean field (MF) message-passing method for an approximate inference. The MF approach tries to approximate the joint posterior  $p(y, h|x)$  by a factorial distribution  $q(y, h) = \prod_{l=1}^C \mu_l^{y_l} (1 - \mu_l)^{1-y_l} \prod_{j=1}^n \tau_j^{h_j} (1 -$

Table 3.1: SRC: Classification Results on REDD

Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
DL	0.4519	0.4983	0.1433
GL	0.5662	0.5839	0.1349
ELM	0.5191	0.5526	0.8884
Proposed	<b>0.6537</b>	<b>0.6801</b>	<b>0.0445</b>

Table 3.2: SRC: Classification Results on Pecan Street

Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
DL	0.6039	0.6049	0.1236
GL	0.6143	0.6206	0.1162
ELM	0.6020	0.6097	0.8989
Proposed	<b>0.7006</b>	<b>0.7035</b>	<b>0.0338</b>

$\tau_j)^{1-h_j}$  that minimizes the Kullback-Leibler (KL) divergence with the true posterior. Running the following message passing procedure to convergence

$$\mu_l \leftarrow \sigma(c_l + \sum_j U_{jl}\tau_j) \quad \forall l \in \{1, \dots, C\}, \quad (3.10)$$

$$\tau_j \leftarrow \sigma(b_j + \sum_b U_{jl}\mu_l + \sum_i W_{ji}x_i) \quad \forall j \in \{1, \dots, n\} \quad (3.11)$$

we can reach a saddle point of the KL divergence, where  $\mu_l$  serves as the estimate for  $p(y_l = 1|x)$  and  $\tau_j$  can be used to estimate  $p(h_j = 1|x)$ .

### 3.2.3 Multi-Label Deep Convolutional Transform Learning

We propose a deeper extension of multi-label convolutional transform learning with a changed cost for the multi-label consistency term. Our justification for

a deep architecture relies on the key property that the solution  $\hat{X}$  to Equation (1.1), assuming fixed filters  $T$ , can be reformulated as the simple application of an element-wise activation function. That is:

$$\operatorname{argmin}_X F(T, X) = \Phi(T \star S), \quad (3.12)$$

with  $\Phi$  being the proximity operator of  $\Psi$  Combettes. It is interesting to remark that, if  $\Psi$  is the indicator function of the positive orthant, then  $\Phi$  identifies with the famous rectified linear unit (ReLU) activation function. Many other examples of mapping between proximity operators and activation functions are provided in Combettes. Consequently, we propose to compute deep features by stacking several such layers, leading to  $X_\ell = \Phi_\ell(T_\ell \star X_{\ell-1})$  with  $\ell = 1, \dots, L-1$  and  $X_0 = S$ .

For both classification and regression tasks, the input remains the same, namely the power consumption over a period of time. For classification task, the labels associated to the data  $S$  are gathered into a matrix  $L$ , where each column is a binary vector with the  $n$ -th element being 0 if the  $n$ -th appliance is off and 1 if the  $n$ -th appliance is on. Owing to such binary nature, it is more appropriate to use a binary cross entropy loss for label consistency, leading to:

$$\begin{aligned} T, X, W \frac{1}{2} \|T_2 \star \Phi(T_1 \star S) - X\|_F^2 + \Psi(X) + \sum_{\ell=1}^2 \left( \mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell) \right) \\ + \eta J_{BCE}(\sigma(WX), L), \end{aligned} \quad (3.13)$$

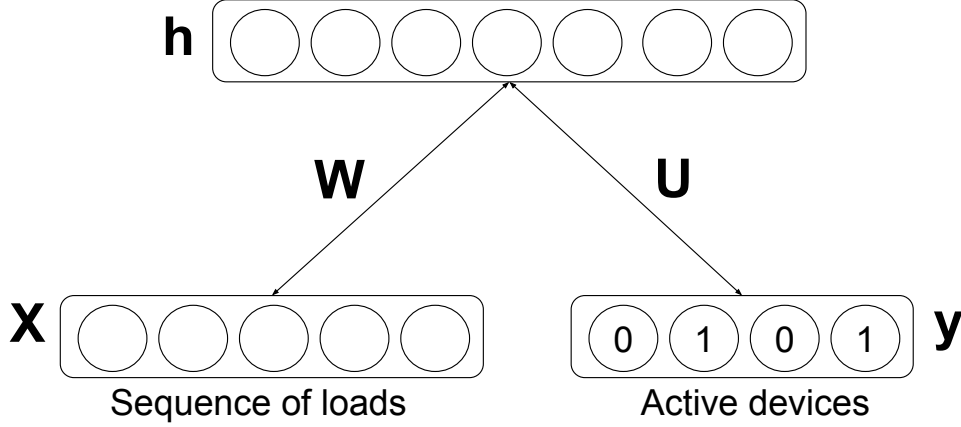


Figure 3.1: Proposed architecture for NILM using multi-label classification RBM.

where  $\sigma$  is the sigmoid function, and  $J_{BCE}$  is the binary cross-entropy loss. Instead of predicting the state of the appliance, if we want to predict the power consumption, the labels in the matrix  $L$  will consist of appliance-wise power consumption. Since the labels will be real values, we use the Euclidean cost, which yields:

$$\begin{aligned}
 T, X, W \frac{1}{2} \|T_2 \star \Phi(T_1 \star S) - X\|_F^2 + \Psi(X) + \sum_{\ell=1}^2 \left( \mu \|T_\ell\|_F^2 - \lambda \log \det(T_\ell) \right) \\
 + \eta \|WX - L\|_F^2.
 \end{aligned} \tag{3.14}$$

Hereabove, we show the formulations for two layers of convolutional transforms ( $T_1$  and  $T_2$ ), but it can be extended to more in a straightforward way. We finally propose to solve both Equations (3.13) and (3.14) using backpropagation with accelerated gradient descent [65].

Figure 3.1 shows the schematic diagram of our proposed method. While it appears to be similar to that of a convolutional neural network (CNN), the key

difference of our proposed approach lies in the way the convolutional filters are learnt. Here, we guarantee uniqueness of the learnt filters, while CNN does not. The later start with random initialization of each filter and ‘hopes’ that the filters will be unique.

### 3.3 Experimental Evaluation

We have carried out experiments on two NILM datasets – REDD and Pecan Street. To emulate real-life scenario for both the datasets aggregated readings over 10 minutes have been considered. We only consider the active power as input and the data for each hour forms the length of the sample. Usually about 70 – 80 percent of the data is used for training the remaining for testing. Our objective is to reduce the required volume of training data, therefore in this work we consider only 10% data for training and 90% for testing. Apart from the ratio of training to test samples, the protocol remains same as [26].

The standard measures for multi-label classification based NILM have been defined in [14]. The  $F1_{macro}$  and the  $F1_{micro}$  are based on the popular  $F1_{score}$  defined for single label classification.

$$F1 = \frac{2 \times TP}{2 \times TP + FN + FP}$$

where  $TP$  is True positive,  $FP$  is false positive and  $FN$  is false negative.

$$F1_{micro} = F1 \left( \sum_{i=1}^N TP_i, \sum_{i=1}^N FP_i, \sum_{i=1}^N FN_i \right)$$

$$F1_{macro} = \frac{1}{N} \sum_{i=1}^N F1(TP_i, FP_i, FN_i)$$

Here,  $TP_i$ ,  $FP_i$  and  $FN_i$  denote the number of true positives, false positive and false negative for the label  $i$ .  $N$  is the number of labels in the dataset.

These measures show how accurately an algorithm can predict the ON / OFF state of appliances. It does not provide insight into the actual energy consumption. For this purpose the second metric defined in [14] is the average energy error (AEE) defined as follows,

$$AEE = \frac{\left| \left( \sum_{i=1}^N Average\_Power_i \right) - \left( \sum_{i=1}^N Actual\_Power \right) \right|}{\sum_{i=1}^N Actual\_Power_i}$$

As mentioned before [26], [52] are the most recent works on multi-label classification based NILM. Both the techniques surpass results from traditional multi-label classification algorithm like MLKNN and RAKEL. The work [26] has shown to improve over other as well as state-of-the-art deep learning techniques like multi-label stacked autoencoder. Therefore in this work we will not compare against the techniques that have already been outperformed by deep learning (DL) [26] and graph learning (GL) [52]. We also compare against the newly developing classification approach of extreme learning machine (ELM); in [66] it has been used multi-label classification.



Table 3.3: ML-SRC: Appliance-Level Evaluation on REDD

Device	DL		GL		ELM		Proposed	
	F1-Score	Error	F1-Score	Error	F1-Score	Error	F1-Score	Error
Dishwasher	0.6409	0.2902	<b>0.6256</b>	0.2516	0.5071	0.9667	<b>0.7433</b>	<b>0.0264</b>
Kitchen Outlet	0.5578	0.2716	0.5071	0.3671	0.6411	0.3326	<b>0.7251</b>	0.9931
Refrigerator	0.5292	0.3628	0.3724	0.5132	0.6118	0.2528	<b>0.7165</b>	0.0373
Washer Dryer	0.3903	0.3122	0.2267	0.6990	0.4977	0.3149	<b>0.7212</b>	0.0911

Table 3.4: ML-SRC: Appliance-Level Evaluation on Pecan Street

Device	MLkNN		RAkEL		LC-DDL		HLM-I layer	
	F1-Score	Error	F1-Score	Error	F1-Score	Error	F1-Score	Error
Air Conditioner	0.6391	0.1720	<b>0.7321</b>	0.8565	0.5882	0.1051	0.7167	0.6785
Dishwasher	0.6546	0.1690	0.7328	0.8490	0.4871	0.1501	<b>0.7376</b>	0.7025
Furnace	0.6123	0.1341	0.7231	0.8415	0.5572	0.0794	<b>0.7297</b>	0.6962
Microwave	0.5916	0.0727	0.6919	0.7301	0.5533	0.0795	<b>0.7189</b>	0.6913

### 3.3.1 Results- Multi-Label Sparse Representation-Based Classification

We compare with three of the latest known tools in multi-label classification – DL, GL and ELM. The overall results are shown in Tables 3.2 and 3.2. We see that for the smaller REDD dataset, DL produces very poor results but for Pecan Street it performs at par with the other benchmarks we have used. This is because REDD is a small dataset and 10% of the data is insufficient for DL and hence it overfits; but Pecan Street is a much larger dataset and 10% of its data is sufficient for the DL to train. Note that even though ELM performs good in terms of  $F1$  measures, the Average Energy Error is very poor.

Of the methods compared against, GL performs the best. It performs reasonably in terms of all the metrics. But ML-SRC yields much better results than GL (and the rest), it is around 10% better in terms of all metrics. For more granular analysis we present the appliance level results for four popular devices in Tables 3.3 and 3.4.

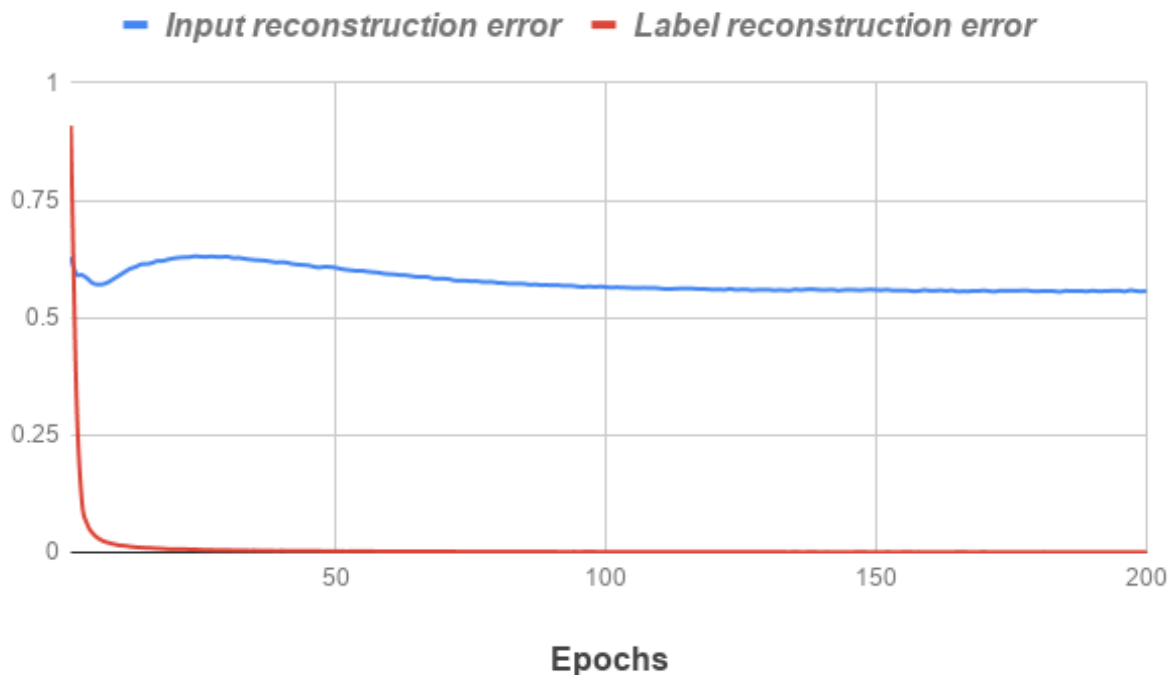


Figure 3.2: Training reconstruction errors of MLC-RBM.

### 3.3.2 Results- Multi-Label Restricted Boltzmann Machine

Table 3.5 and 3.6 present the F1-Score and correspondingly obtained disaggregation error for each target device in both the datasets. Table 3.7 and 3.8 contain micro and macro F1-Scores yielded by the state-of-the-art and proposed algorithm on the REDD and Pecan Street dataset respectively. Our proposed model yields the best results regarding classification measures and gives comparable disaggregation accuracy. Although best classification accuracy should reflect the least disaggregation error, here it is not so. This mismatch engenders an ambiguity in results.

We would like to clarify it with an example. Suppose true labels for two hours of aggregate consumption of four devices are 1 0 0 1 and 0 1 1 0 whereas

the predicted labels are 0 1 1 0 and 1 0 0 1 respectively. For the given case F1-Score would be zero as all the identified states are wrong. For the same case, disaggregation accuracy would be 100 % as the number of identified active appliances exactly matches the number of true active appliances. This example explains why techniques, such as Label-Consistent Deep Dictionary Learning (LC-DDL) [26], gives the best disaggregation accuracy but worst F1-Scores. Therefore in such a framework, the performance of an algorithm should be judged only after looking at both metrics collectively.

Table 3.5: MLC-RBM: Appliance-Level Evaluation on REDD

Device	MLkNN		RAkEL		LC-DDL		MLC-RBM	
	F1-Score	Error	F1-Score	Error	F1-Score	Error	F1-Score	Error
Lighting	0.6476	0.3718	0.6760	0.8213	0.6216	0.2608	<b>0.6947</b>	<b>0.1762</b>
Kitchen	0.5081	0.4304	0.6108	0.6995	0.6411	0.3326	<b>0.7213</b>	<b>0.1273</b>
Refrigerator	0.5292	0.3628	0.6724	0.5132	0.6118	0.2528	<b>0.7186</b>	<b>0.1644</b>
Washer Dryer	0.3903	0.3122	0.5267	0.6990	0.4977	0.3149	<b>0.6983</b>	<b>0.1963</b>

Table 3.6: MLC-RBM: Appliance-Level Evaluation on Pecan Street

Device	MLkNN		RAkEL		LC-DDL		MLC-RBM	
	F1-Score	Error	F1-Score	Error	F1-Score	Error	F1-Score	Error
Air Conditioner	0.6391	0.1720	0.6521	0.8565	0.5882	<b>0.1051</b>	<b>0.7023</b>	0.2334
Dishwasher	0.6546	0.1690	0.6728	0.8490	0.4871	0.1501	<b>0.7269</b>	<b>0.1341</b>
Furnace	0.6123	0.1341	0.6231	0.8415	0.5572	<b>0.0794</b>	<b>0.7113</b>	0.2224
Microwave	0.5916	<b>0.0727</b>	0.6819	0.7301	0.5533	0.0795	<b>0.6981</b>	0.1985

Table 3.7: MLC-RBM: Performance Evaluation on REDD

Method	Macro F1-Score	Micro F1-Score
MLkNN	0.6086	0.6143
RAkEL	0.6290	0.6294
LC-DDL	0.5222	0.5262
MLC-RBM	<b>0.7082</b>	<b>0.7157</b>

Table 3.8: **MLC-RBM: Performance Evaluation on Pecan Street**

Method	Macro F1-Score	Micro F1-Score
MLkNN	0.6183	0.6194
RAkEL	0.5872	0.6019
LC-DDL	0.5214	0.5332
MLC-RBM	<b>0.7080</b>	<b>0.7123</b>

### 3.3.3 Multi-Label Deep Convolutional Transform Learning

#### 3.3.4 Classification

In the multi-label classification scenario, we have compared with two state-of-the-art techniques, namely deep learning based NILM with pinball loss (PB-NILM) [67] and multi-label deep transform learning (MLDTL) [26]. For our proposed technique the parameters were determined using k-fold cross validation on the training data. The retained parameters were  $\beta = 1$ ,  $\mu = 3$ ,  $\lambda = 1$  and  $\eta = 1$ . The  $F1_{macro}$ , the  $F1_{micro}$  and the average energy error (AEE) are presented in Table 3.9 and 3.10.

For both the datasets, we see that our proposed algorithm with two layers performs the best. Adding further layers on these small datasets results in over-fitting. We find that PB-NILM works better for Pecan Street (larger dataset) compared to REDD which may owe to the fact that the technique is over-fitting for the smaller data.

Table 3.9: **Deep-CTL: Classification Results on REDD**

Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
PB-NILM	0.5515	0.5576	0.3903
ML-DTL	0.5693	0.5642	0.3537
Proposed 1 layer	0.5687	0.5682	0.2926
Proposed 2 layer	<b>0.6018</b>	<b>0.6026</b>	<b>0.2558</b>
Proposed 3 layer	0.5425	0.5419	0.3282

Table 3.10: **Deep-CTL: Classification Results on Pecan Street**

Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
PB-NILM	0.6231	0.6207	0.2582
ML-DTL	0.5552	0.5552	0.4048
Proposed 1 layer	0.6121	0.6119	0.2723
Proposed 2 layer	<b>0.6381</b>	<b>0.6378</b>	<b>0.2316</b>
Proposed 3 layer	0.5983	0.5963	0.2902

### 3.3.5 Regression

In this scenario, our objective is to predict the energy consumed by different appliances. We have compared against deep latent generative model (DLGM) [68] and semi-binary non-negative matrix factorization (SMNMF) [69]. The parameters for the existing benchmarks have been obtained from the papers. The parametric values for our model remain the same as before. For comparing the accuracies, we compute the normalized energy errors of common appliances for the two different datasets. The results are shown in Tables 3.7 and 3.8. Here we are showing the best results from our two layer architecture.

### 3.4 Discussion

In recent times, there is a concerted effort towards truly non-intrusive load monitoring [14, 26, 52, 70]. This is required for practical large scale roll-out of NILM as a service with the larger goal of improving energy sustainability. In this respect the multi-label classification framework has been showing promise [14, 26, 52]. However, recent deep learning based solutions like [26] require large volume of labeled training data. In order to reduce that requirement we propose a simple solution based on adapting the SRC framework [53] to solve multilabel classification problems. Our proposed multi-label SRC improves over state-of-the-art techniques by a considerable margin.

The main shortcoming of our approach is that, it is not possible to estimate different stages of each appliance via our method. Neural network based function approximation approaches may be better in this respect. SRC is the basic algorithm. Over the years various modification have been proposed, such as kernel SRC [71] group SRC [72], dictionary learnt SRC [73] etc. We propose to adapt all the popular variants to solve multi-label classification problems.

Next, we propose multi-label classification RBM. RBM has good reconstruction ability and when combined with multi-label supervision also provides good classification accuracy. We compare the proposed technique with all the prior works where NILM was transformed as a multi-label classification task. Our proposed model yields the best results in term of classification accuracy and comparable results regarding energy disaggregation.

Last, we propose a new supervised deep learning framework. It is based on the concept of convolutional transform learning. It can tackle both to classification problem for identifying the states of the appliances and to the regression problem for estimating their energy consumptions. Comparisons with state-of-the-art techniques show that our proposed method improves over the rest. We expect that the results can be further improved by adopting post-processing approaches such as [69].

## Chapter 4

# Blind Compressed Sensing for Non-Intrusive Load Monitoring

Energy disaggregation is a single channel (smart-meter) blind source (appliances) separation problem. This makes the problem highly underdetermined in nature – one equation (smart-meter consumption) and many variable (appliance consumption). Therefore the problem has infinitely many solutions.

Smart-meters can sample at high frequencies, but higher frequencies mean generation of more data. Transmitting this data from the building smart-meter to the cloud at the utilities consumes some bandwidth; higher the sampling frequency higher is the bandwidth consumed. Note that, it is not only one building that would be transmitting this data, all the buildings would be transmitting it; in such a scenario it is likely the entire network bandwidth will be consumed for only transmitting power signals! To keep the network usage at check, the smart-meter transmits the signal at low-frequencies (even though it is capable of sampling at high frequencies).



Typically it is expected that energy disaggregation would be offered as a service by the utilities. However, since the utilities will have access to only low-frequency information, the disaggregation accuracy is likely to suffer. To bridge the gap between high frequency sampling and low-frequency transmission we propose a compressed sensing / compressive sampling (CS) approach [74–76].

There are two techniques presented in this chapter-1) Blind Compressed Sensing and 2) Multi-Label Deep Blind Compressed Sensing. The first one presents a solution to recover the high frequency energy signals from the CS samples. In the second one, from such CS samples, we propose to detect the state of the appliance by using a multi-label consistent version of deep blind compressed sensing.

The chapter is organized into several sections. We will discuss the basics of CS in the Section 4.1. In Section 4.2, we describe our proposed formulation. The results will be detailed in Section 4.4. Finally, the conclusions of this work will be discussed in Section 4.6.

## **4.1 Literature Review**

When the power signal is of sufficiently high frequency, integer programming based approaches provide a feasible solution [77,78]. Similarly factorial hidden Markov model (FHMM) is used to disaggregate appliance loads from high frequency samples [19,48]. The performance of such techniques degrades when the sampling frequency is reduced. Sparse coding approaches yield somewhat

better results at low-frequencies [12, 13]; however, even with sparse coding, higher frequencies translate to better results. We project the high frequency signal to a lower dimension embedding by a random projection matrix. The lower dimensional signal will emulate a low frequency signal which can be then transmitted.

The random projection can be easily integrated into hardware [79, 80]. Under certain conditions, such a lower dimensional embedding approximately preserves the information of the high frequency signal and can be recovered using sparsity promoting techniques like  $l_1$ -minimization [81] or matching pursuits like algorithms [82].

This work extends the traditional compressed sensing dictates (i.e., recovering the signal) by adding simultaneous disaggregation as part of the recovery process. Our formulation is based on the dictionary learning approach [83] (the same technique used in sparse coding [12, 13]).

#### 4.1.1 Blind Compressed Sensing

Compressed Sensing (CS) studies the problem of solving an underdetermined linear system of equations where the solution is known to be sparse. In practical scenarios, the system is corrupted by noise as well.

$$y_{m \times n} x_{n \times 1} + \varepsilon_{m \times 1}, m < n \quad (4.1)$$

The solution  $x$  is assumed to be  $k$  sparse ( $k \ll m < n$ ). For an under-determined system, there can be infinitely many solutions. Research shows that when the solution is sparse, it is necessarily unique [84]; i.e., there cannot be more than one sparse solution. Further research established that when the number of equations ( $m$ ) satisfies the following criterion (Equation (4.2)),  $l_1$ -minimization can recover the sparse solution.

$$m = ck \log\left(\frac{n}{k}\right) \quad (4.2)$$

here  $c$  is a positive constant. The  $l_1$ -norm minimization is robust to noise [85].

The recovery is formulated as:

$$\min_x \|y - Ax\|_2^2 + \lambda \|x\|_1 \quad (4.3)$$

CS recovery is not possible for any system of equations  $A$ ; it is only guaranteed when the so called restricted isometric property (RIP) holds. This condition is expressed as follows:

$$(1 - \delta) \|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta) \|x\|_2^2 \quad (4.4)$$

Here  $\delta$  is a small constant. RIP guarantees that the system  $A$  behaves as a near isometry. The value of  $\delta$  dictates how much the system deviates from ideal Isometry. This property is usually satisfied by random matrices for example, restricted Fourier ensembles and matrices drawn from distributions such as Gaussian, Bernoulli, and Binomial.

Practical systems/signals are hardly ever sparse. However, most of them have a sparse representation in some transform domain. For example, images are sparse in discrete cosine transform (DCT) or wavelet, speech is sparse in short time Fourier transform, etc. This phenomenon allows expression of the signal  $x$  in terms of transform domain sparse coefficients  $\alpha$ ,

$$\textit{Analysis} : \alpha = \psi x \quad (4.5a)$$

$$\textit{Synthesis} : x = \psi^T \alpha \quad (4.5b)$$

Here  $\psi$  is the sparsifying transform and the relationships (Equation (4.5)) hold for orthogonal ( $\psi^T \psi = I = \psi \psi^T$ ) and ( $\psi^T \psi = I \neq \psi \psi^T$ ) tight-frame systems.

For signals that have a sparse representation in the transform domain, the recovery is expressed as follows:

$$\min_{\alpha} \|y - A\psi^T \alpha\|_2^2 + \lambda \|\alpha\|_1 \quad (4.6)$$

Once the sparse coefficients are recovered, the signal is obtained by applying the synthesis Equation (4.5b).

Following Equation (4.2), note the number of equations/samples needed to recover a signal is directly dependent on the sparse representation and thereby on the choice of the transform  $\psi$ . For example, if is an image, the number of corresponding non-zero DCT coefficients will be higher than the corresponding wavelet coefficients making the choice of transform a crucial step in CS recovery.

Fixed transforms like Fourier, DCT and Wavelet have nice mathematical properties but are not known to produce the sparsest representation. In signal processing, it is well known that an adaptive basis (learnt from the signal) produces the sparsest representations. This paved the way for dictionary learning based solutions; starting with the work on K-SVD [86].

The idea of blind compressed sensing (BCS) was introduced by Gleichman and Eldar [87]; it married dictionary learning with compressed sensing. In BCS, the sparsity basis is learnt from the data (also known as dictionary learning). For example, if the problem involves an image, the sparsity basis is learnt from the patches of the image. The recovery is posed as:

$$\min_{x, D, z_i's} \|y - Ax\|_2^2 + \mu \left( \underbrace{\sum_i \|P_i x - Dz_i\| + \lambda \|z_i\|_1}_{\text{Dictionary Learning}} \right) \quad (4.7)$$

Here  $P_i$  representation patch extraction operator;  $D$  is the basis that is being adaptively learnt from the patches and  $z_i$  are the corresponding sparse representations of the patch  $P_i x$ . In dictionary learning,  $D$  replaces the role of  $\psi$  in CS.

When the basis is learnt adaptively (i.e. in case of BCS) the recovery results are far better than of classical CS where the sparsifying basis is fixed. There are many other branches of CS and dictionary learning, but these are not pertinent to us. The interested reader may peruse [88].

BCS was extended to a deeper formulation [89] by incorporating deep dic-

tionary learning [90] into the CS.

## 4.2 Proposed Formulation for Blind compressed NILM

We assume that the smart-meter is sampling at the rate of  $n$  samples per unit of time (say an hour); but is only allowed to transmit  $m < n$  samples in that period. Let  $x_{n \times 1}$  represent the signal sampled by the smart-meter. Currently a sub-sampled version of  $x$  is transmitted; we propose to embed the high dimensional signal into a lower dimensional representation  $y_{m \times 1}$  by a random projection matrix  $A_{m \times n}$  (satisfying RIP). This is represented by,

$$y = Ax + \epsilon \quad (4.8)$$

It is unlikely that the system will be corrupted by noise, but for the sake of generality, we assume Gaussian noise  $\epsilon$ . The problem is to disaggregate the appliance level consumption given the lower dimensional representation  $y$ . To do so, a standard NILM training and testing approach is followed.

### 4.2.1 Training

In the training phase the individual appliances are metered. For each appliance  $j$ , the samples for the  $i^{th}$  unit of time is represented by  $x_i^j$ . The complete training data for the  $j^{th}$  appliances is represented by,

$$x^j = \left[ x_1^j | x_2^j | \dots | x_N^j \right] \quad (4.9)$$

Here we assume  $N$  units of time in the training phase. In our proposition, the utilities do not have access to  $X^j$ , but has received a lower dimensional projection of it, given by:

$$Y^j = AX^j + E^j \quad (4.10)$$

where  $y^j = [y_1^j | y_2^j | \dots | y_N^j]$  and  $x^j = [\epsilon_1^j | \epsilon_2^j | \dots | \epsilon_N^j]$ .

Following the work of sparse coding [12], each appliance is modelled by a sparse codebook/dictionary  $D^j$ . This is expressed as,

$$X^j = D^j Z^j \quad (4.11)$$

We reiterate that our work assumes that the disaggregation happens at the utilities server/cloud. Incorporating this model (Equation (4.11)) to the data received at the utilities we get:

$$Y^j = AD^j Z^j + E^j \quad (4.12)$$

Following the work on sparse coding, the training phase requires solving for the dictionaries and the sparse codes (not required during testing) for modeling the appliances. This is expressed as,

$$\min_{D^j, Z^j} \|Y^j - AD^j Z^j\|_F^2 + \lambda \|Z^j\|_1 \quad (4.13)$$

Equation (4.13) is easily solved using alternating minimization of the codebook and the sparse codes. During the update for the codebook, the sparse code is assumed to be constant. The update is given by:

$$\min_{D^j} \|Y^j - AD^j Z^j\|_F^2 + \lambda \|Z^j\|_1 \Rightarrow D^j = A^\dagger Y^j (Z^j)^\dagger$$

where  $(.)^\dagger$  denotes the Moore-Penrose pseudo inverse. The update for the sparse codes assumes that the codebook is fixed. The update is expressed as,

$$\min_{Z^j} \|Y^j - AD^j Z^j\|_F^2 + \lambda \|Z^j\|_1 \quad (4.14)$$

This is a standard  $l_1$ -minimization problem that can be solved using any iterative thresholding algorithm. Note that the solution to the codebook and sparse codes automatically reconstructs the original signals acquired by the smart-meter Equation (4.11).

#### 4.2.2 Testing

In the testing/operation stage, the task is to disaggregate the total load acquired by the smart-meter. The total load, as recorded by the smart-meter, has a unit of time. Therefore for all units of time, the data is expressed as,

$$x^j = [x_1|x_2|\dots|x_M] \quad (4.15)$$

Equation (4.15) is an aggregate of the loads consumed by individual appliances.

$$X = \sum_j X^j \quad (4.16)$$

Incorporating the sparse coding model (Equation (4.11)) into (Equation (4.16))



leads to:

$$X = \sum_j X^j = \sum_j D^j Z^j \quad (4.17)$$

As mentioned before, the sparse codes obtained during the training phase are not useful later on, only the codebooks are used in Equation (4.17). In the compressive NILM scenario, the utilities do not have access to the fully sampled data  $X$ , but has received its lower dimensional embedding  $Y : Y = AX + E$ . Incorporating Equation (4.17) into the data acquisition model leads to:

$$Y = A \sum_j D^j Z^j + E \quad (4.18)$$

During the testing phase, the codebooks are known; the goal is to estimate the sparse codes  $Z^j$ . The solution is obtained by minimizing the following,

$$\min_{Z^j's} \left\| Y - A \sum_j D^j Z^j \right\|_F^2 + \sum_j \|Z^j\|_1 \quad (4.19)$$

As before, this can be solved using any iterative thresholding algorithm.

Once the sparse codes are solved, the power consumption of individual devices can be obtained using Equation (4.11). Note that our algorithm automatically reconstructs the signal during training and testing phase. If one is interested in applying some other algorithm, they can run it on the reconstructed data at the server/cloud.

### 4.3 Proposed formulation for Deep Blind Compressed NILM

The model is expressed as –

$$Y = AD_1D_2D_3Z + N \quad (4.20)$$

In deep BCS, instead of learning a single layer of dictionary multiple layers –  $D_1$ ,  $D_2$  and  $D_3$ , are learnt. The solution to Equation (4.20) was posed as –

$$\min_{D_1, D_2, \dots, D_N, Z} \|X - D_1D_2\dots D_NZ\|_F^2 + \lambda_1\left(\sum_i \|D_i\|_F^2\right) + \lambda_2\|Z\|_1 \quad (4.21)$$

The objective of deep BCS was not signal estimation, but unsupervised feature extraction ( $Z$ ) from compressively sampled data ( $Y$ ).

The unsupervised deep BCS model of [89] was later supervised in [91] by adding a label consistency term.

$$\min_{D_1, D_2, \dots, D_N, Z} \|X - D_1D_2\dots D_NZ\|_F^2 + \lambda_1\left(\sum_i \|D_i\|_F^2\right) + \lambda_3\|L - WZ\|_F^2 \quad (4.22)$$

Here  $L$  is the one hot encoded label vectors and  $W$  the linear classifier that projects the coefficients / features ( $Z$ ) onto the space of binary labels  $L$ . Note that in [91] the  $l_1$ -norm on the coefficients is dropped. It was argued that the sparsity promoting norm is necessary for solving synthesis problems in CS but carries no meaning in analysis tasks. Furthermore, as the number of basis elements are progressively reduced in deep BCS, the coefficient is small; constraining the coefficients by an  $l_1$ -norm would hamper its representation ability. The

training phase is accomplished by solving Equation (4.22).

During testing, assuming that  $x_{test}$  is the hourly high frequency input data, the observed low-frequency data is  $y_{test}$ . Given the observation and the projection operator  $A$ , one can find the representation  $z_{test}$  by solving the following,

$$\min_{z_{test}} \|y_{test} - AD_1D_2D_3z_{test}\|_F^2 \quad (4.23)$$

Note that the dictionaries are fixed in Equation (4.23), they are learnt during the training phase. Once  $z_{test}$  has been generated by solving Equation (4.23), the class label vector is obtained by

$$l_{test} = Wz_{test} \quad (4.24)$$

The label vector thus obtained is unlikely to be binary; in most cases, it will contain real values at all the positions. To form the binary vector it needs to be thresholded. In this work, we have assumed an empirical threshold of 0.5, i.e. at positions having values greater than 0.5 we are assigning it to be 1, and at positions where the values are less than or equal to 0.5 we are assigning it to be 0. From the thus thresholded vector of binary values, one can obtain the state of the appliance during the test hour.

Note that as a by-product of our technique one can get the recovered high-frequency data at the server from  $\hat{x} = D_1D_2D_3z_{test}$ . It is possible to run third-party disaggregation algorithms on the recovered data.

Table 4.1: cNILM: Disaggregation Performance Evaluation (Using Precision/Recall)

Appliance	Sub-Sampled SC	Sub-Sampled FHMM	Blind Compressive NILM	Reconstructed FHMM
Microwave	.53/.34	.55/.32	.70/.58	.71/.55
Kitcehn Outlet1	.30/.11	.27/.10	.37/.15	.35/.13
Kitchen Outlet 2	.33/.11	.32/.11	.45/.15	.40/.14
Furnace	.75/.61	.78/.59	.86/.69	.87/.66
Washer/Dryer	.73/.54	.74/.53	.82/.64	.85/.62

For this work, we learnt three dictionaries because the results saturated after this. One can have a fewer or larger number of dictionaries in other applications. The optimization problems and the corresponding solutions will follow directly from the treatment shown herein.

#### 4.4 Experimental Evaluation of BCS

Here we report results on the REDD dataset. We assume that the utilities acquire the data once every minute. Given this constraint, we further sub-sample the data to once every ten minutes (sub-sampled data), from 60 samples per hour. In the proposed compressive NILM regime, the once per minute data is projected to a lower dimension of 6 samples in hour using a Bernoulli matrix (compressively sampled data); using a Bernoulli projection matrix ensures that our simulations are hardware friendly.

To compare the performance of our proposed approach, we employ FHMM [19] and sparse coding (SC) [12] on the sub-sampled version of the data. These act as the benchmarks. We use our algorithm to disaggregate from the compressive sampled version of the data. As mentioned before, our proposed method

Table 4.2: Deep-cNILM: Classification Results on REDD

Type of Sampling	Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
CS	cNILM	0.5593	0.5515	0.1337
	Deep-cNILM	0.6110	0.6115	0.1219
TS	SC	0.4754	0.4760	0.0964
	FHMM	0.4555	0.4572	0.1286
	LFNILM	0.5508	0.5513	0.1310

reconstructs the high frequency data in the process; on this reconstructed data we apply FHMM. Note that, there is no point in applying sparse coding on the reconstructed data, since our proposed approach effectively does the same.

Appliance-level Precision and Recall are used as metrics for evaluating the performance [26]. While using such a metrics, we tacitly assume that the appliances are binary-state (ON or OFF) while disregard other operational states.

The results shown in Table 4.1 depict the expected trend. Both the techniques SC and FHMM perform similarly for a given sampling frequency. The performance is poor when the sampling is done at regular intervals (sub-sampled data); but with compressive sampling, the performance improves considerably. Blind Compressive NILM effectively disaggregates (using SC) and reconstructs simultaneously. The FHMM is applied on the thus reconstructed (higher frequency – once very minute) data.

#### 4.5 Experimental Evaluation of Multi-Label BCS

We have carried out experiments on REDD and Pecan Street dataset. High frequency data is available for both the datasets. We consider two scenarios –

Table 4.3: Deep-cNILM: Classification Results on Pecan Street

Type of Sampling	Method	Macro F1 Measure	Micro F1 Measure	Average Energy Error
CS	cNILM	0.5384	0.5393	0.2559
	Deep-cNILM	0.5961	0.5964	0.2262
TS	SC	0.5148	0.5148	0.1061
	FHMM	0.4827	0.4832	0.1594
	LFNILM	0.5366	0.5370	0.2495

traditional sampling (TS) vs compressive sampling (CS). In TS, to emulate real-life scenario for both the datasets aggregated readings over 15 minutes have been considered; the corresponding play-out rates are 60:15. For CS, we keep the play-out rate equivalent to that of TS, i.e. once every 15 minutes would correspond to 60:15. For each house, 60% of the samples are used for training and the remaining 40% for testing in chronological order.

For the CS scenario we only have two techniques, the BCS NILM and deep-cNILM. In TS, we compare with two standard benchmarks for NILM, viz. factorial hidden markov model(FHMM) and sparse coding (SC); SC has shown to yield good results on low frequency measurements empirically even though it was not specifically meant for this task. We have also compared with a low frequency NILM (LFNILM) technique [92].

The standard measures for multi-label classification based NILM- the  $F1_{macro}$ , the  $F1_{micro}$  and average energy error (AEE) are same as defined in the previous chapter.

For the REDD dataset two layers of dictionaries are used and for Pecan Street three layers. This is because REDD is a relatively small dataset and adding

Table 4.4: Comparison of Runtimes in Seconds

Method	cNILM	Deep-cNILM	SC	FHMM	LFNILM
Runtime	406	118	367	411	506

Table 4.5: Deep-cNILM: Classification Results on Pecan Street

REDD	Pecan Street	Measure		
		Measure	Measure	Energy Error
CS	cNILM	0.5384	0.5393	0.2559
	Deep-cNILM	0.5961	0.5964	0.2262
TS	SC	0.5148	0.5148	0.1061
	FHMM	0.4827	0.4832	0.1594
	LFNILM	0.5366	0.5370	0.2495

more layers tend to overfit. For the same reason, adding more layers to the Pecan Street experiments deteriorated the results. For both datasets, we follow the thumb rule of halving the number of dictionary atoms in subsequent layers.

Our approach requires the specification of two parameters -  $\lambda_1$  and  $\lambda_3$ . The parameter  $\lambda_3$  controls the relative importance of the reconstruction and classification penalties. Since there is no reason to favor one over the other, we keep it at unity. The value of  $\lambda_1=0.05$  has been used throughout. We found that the results are stable between the values 0.01 and 0.1.

We find that Deep-cNILM always performs the best, followed by cNILM and LFNILM. SC and FHMM yields the worst results (Table 4.2 and 4.5). The results are at par with existing literature. It is known from Table 4.1 that cNILM

Table 4.6: Comparison of Reconstruction

REDD				Pecan Street			
Proposed		cNILM		Proposed		cNILM	
Train	Test	Train	Test	Train	Test	Train	Test
0.1052	0.1667	0.1286	0.1412	0.1101	0.1713	0.1309	0.1458

yields somewhat better results than SC and FHMM. The custom LFNILM is almost at par with cNILM. However, our method yields the best results.

We have shown the run-time comparison on the test data in Table 4.4. All the algorithms have been run on MATLAB 2019 on a PC having 8GB of RAM running 64 bit Windows. The average runtimes are shown in seconds for each house per day – combining REDD and Pecan Street. We see that our proposed method is the fastest. This is because the problem that needs to be solved during runtime, Equation (4.23) and Equation (4.24), are computationally cheap. Both SC and cNILM require solving relatively complex optimization problems ( $l_1$ -minimization) and hence are more time consuming. The standard FHMM is known to take longer than SC. LFNILM is an engineered solution that is relatively the slowest.

We mentioned that our methods recovers the original signal as a by-product. To test the quality of recovery, we compute the normalized mean squared error (NMSE) between the actual and the reconstructed. This is defined as:

$$\frac{\|x_{actual} - x_{reconstructed}\|_2}{\|x_{actual}\|_2} \quad (4.25)$$

The results for testing and training data are shown in Table ???. We find that the recovery of training data is considerably better than that of test data for both the proposed and cNILM techniques. However cNILM results are more balanced than ours. This is because ours is built on deep dictionary learning, and the multiple dictionaries tend to overfit the train data and loses its generalization



ability on test data. We could have a more balanced recovery by changing the parameters of the algorithm, but that would have affected the quality of disaggregation. We did not compromise on that end. The existing method cNILM is shallow and does not overfit as much.

## 4.6 Discussion

This is the first work (proof-of-concept) on the topic of employing compressed sensing to balance accuracy and bandwidth for NILM tasks. The results show that the approach is promising. In the future we would like to extend the work in two ways. First, we would like to push the extents of the compressive sampling and attempt to disaggregate and reconstruct from more aggressive sampling (compared to 10:1 used in this work). Second, we would like to attempt some state-of-the-art deep learning models into the CS framework to improve the disaggregation results.

## Chapter 5

# Conclusion

This thesis addressed several bottlenecks (discussed the chapters) in the area of non-intrusive load monitoring. We have tried to propose solutions to overcome these bottlenecks to make the process more practical and scalable.

### 5.1 Summary of Contribution

The thesis contributions are outlined below:

- We propose Deep Sparse Coding (DSC) and Analysis Co-sparse Coding algorithms for NILM. DSC is a deep learning approach—instead of learning one level of dictionary, we learn multiple layers of dictionaries for each device. The experiment show that the proposed algorithm beats the existing benchmark in terms of disaggregation accuracy.

Further, Analysis Co-Sparse Coding requires only 10% of the total data for training as compared to 50% of the total data required by the bench-

mark techniques. This method gives at par accuracy with the pre-existing methods. It means that we can save sensor installment cost and reduce the duration of intrusiveness during data acquisition phase.

- Next, we follow the approach of multi-label classification for non-intrusive load monitoring (NILM). We modify the popular sparse representation based classifier, Restricted Boltzmann Machine, and Deep Convolutional Transform Learning (developed for single label classification) to solve multi-label classification problems. Results on benchmark datasets i.e. REDD and Pecan Street datasets show significant improvement over state-of-the-art techniques.
- Last, we propose a compressive sampling (CS) approach. The high-frequency power signal measurements from a smart meter are encoded (by a random matrix) to a very few samples making the signal suitable for WAN transmission without choking network bandwidth. CS guarantees the recovery of the high-frequency signal from the few transmitted samples under certain conditions (discussed in Chapter 4). The work shows how to simultaneously recover the signal and disaggregate it using two approaches- direct and via multi-label classification approach.

## **5.2 Future Work**

In this section, we discuss the further advancements of the thesis work.

### **5.2.1 Domain Adaptation**

A common problem with the NILM training phase is that one has to collect the data whenever disaggregation needs to be performed in a new building. This significantly increases the overall cost.

To overcome the challenge, we can explore the idea of domain adaptation/transfer learning in NILM. There are two ways to do it. First, we can further tune the already learnt weights on one building by using small-sized data from a new building. Second, we can augment small-sized data from the new building with another building's data containing the same devices and then train the model with the augmented data.

### **5.2.2 Training-less Non-Intrusive Load Monitoring**

We can further look into the training-less NILM owing to its various advantages [70]. This approach does not require any prior information about the house or appliances. The performance of this method does not get impacted by the addition or removal of appliances.

In training-less NILM, first windows of events are figured out. These windows indicate that one or more appliances have changed their states. After extracting features from these windows, event detection is performed to identify the participating devices.

## References

- [1] S. Hayes and C. Kubes, “Saving energy, saving lives: The health impacts of avoiding power plant pollution with energy efficiency,” in *American Council for an Energy-Efficient Economy*, 2018.
- [2] D. Archer, *Global warming: understanding the forecast*. John Wiley & Sons, 2011.
- [3] D. J. Nowak, N. Appleton, A. Ellis, and E. Greenfield, “Residential building energy conservation and avoided power plant emissions by urban and community trees in the united states,” *Urban Forestry & Urban Greening*, vol. 21, pp. 158–165, 2017.
- [4] S. Darby *et al.*, “The effectiveness of feedback on energy consumption,” *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, vol. 486, no. 2006, p. 26, 2006.
- [5] G. W. Hart, “Nonintrusive appliance load monitoring,” *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992.
- [6] J. O. Pérez-Lombard, Luis and C. Pout, “A review on buildings energy consumption information,” in *Energy and buildings*. Elsevier, 2008.

- [7] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel, “Disaggregated end-use energy sensing for the smart grid,” *IEEE pervasive computing*, vol. 10, no. 1, pp. 28–39, 2010.
- [8] T. Babaei, H. Abdi, C. P. Lim, and S. Nahavandi, “A study and a directory of energy consumption data sets of buildings,” *Energy and Buildings*, vol. 94, pp. 91–99, 2015.
- [9] M. Dong, P. C. Meira, W. Xu, and C. Chung, “Non-intrusive signature extraction for major residential loads,” *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1421–1430, 2013.
- [10] J. Paris, J. S. Donnal, and S. B. Leeb, “Nilmdb: The non-intrusive load monitor database,” *IEEE Transactions on Smart Grid*, vol. 5, no. 5, pp. 2459–2467, 2014.
- [11] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, “Unsupervised disaggregation of low frequency power measurements,” in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 747–758.
- [12] J. Z. Kolter, S. Batra, and A. Y. Ng, “Energy disaggregation via discriminative sparse coding,” in *Advances in Neural Information Processing Systems*, 2010, pp. 1153–1161.
- [13] E. Elhamifar and S. Sastry, “Energy disaggregation via learning powerlets and sparse coding,” in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2015)*, 2015.

- [14] S. M. Tabatabaei, S. Dick, and W. Xu, “Toward non-intrusive load monitoring via multi-label classification,” *IEEE Transactions on Smart Grid*, vol. 8, no. 1, pp. 26–40, 2017.
- [15] D. Li and S. Dick, “Whole-house non-intrusive appliance load monitoring via multi-label classification,” in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 2749–2755.
- [16] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura, and K. Ito, “Nonintrusive appliance load monitoring based on integer programming,” in *SICE Annual Conference, 2008*. IEEE, 2008, pp. 2742–2747.
- [17] M. Nguyen, S. Alshareef, A. Gilani, and W. G. Morsi, “A novel feature extraction and classification algorithm based on power components using single-point monitoring for nilm,” in *Electrical and Computer Engineering (CCECE), 2015 IEEE 28th Canadian Conference on*. IEEE, 2015, pp. 37–40.
- [18] O. Parson, S. Ghosh, M. Weal, and A. Rogers, “Using hidden markov models for iterative non-intrusive appliance monitoring,” 2011.
- [19] J. Z. Kolter and T. Jaakkola, “Approximate inference in additive factorial hmms with application to energy disaggregation,” in *Artificial Intelligence and Statistics*, 2012, pp. 1472–1482.
- [20] M. Collins, “Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms,” in *Proceedings of the ACL-02 conference on Empirical methods in natural language*

- processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 1–8.
- [21] D. Srinivasan, W. Ng, and A. Liew, “Neural-network-based signature recognition for harmonic source identification,” *IEEE Transactions on Power Delivery*, vol. 21, no. 1, pp. 398–405, 2006.
- [22] J. Kelly and W. Knottenbelt, “Neural nilm: Deep neural networks applied to energy disaggregation,” in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM, 2015, pp. 55–64.
- [23] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.
- [24] G. Tsoumakas and I. Vlahavas, “Random k-labelsets: An ensemble method for multilabel classification,” in *European conference on machine learning*. Springer, 2007, pp. 406–417.
- [25] M.-L. Zhang and Z.-H. Zhou, “A k-nearest neighbor based algorithm for multi-label classification,” in *Granular Computing, 2005 IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 718–721.
- [26] V. Singhal, J. Maggu, and A. Majumdar, “Simultaneous detection of multiple appliances from smart-meter measurements via multi-label consistent deep dictionary learning and deep transform learning,” *IEEE Transactions on Smart Grid*, 2018.



- [27] J. Z. Kolter and M. J. Johnson, “Redd: A public data set for energy disaggregation research,” in *Workshop on Data Mining Applications in Sustainability (SIGKDD)*, San Diego, CA, vol. 25, no. Citeseer, 2011, pp. 59–62.
- [28] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, “Nilmtk: an open source toolkit for non-intrusive load monitoring,” in *Proceedings of the 5th international conference on Future energy systems*. ACM, 2014, pp. 265–276.
- [29] R. Rubinstein, T. Peleg, and M. Elad, “Analysis k-svd: A dictionary-learning algorithm for the analysis sparse model,” *IEEE Transactions on Signal Processing*, vol. 61, no. 3, pp. 661–677, 2012.
- [30] M. Aharon, M. Elad, and A. Bruckstein, “K-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.
- [31] W. Ruangsang and S. Aramvith, “Super-resolution for hd to 4k using analysis k-svd dictionary and adaptive elastic-net,” in *2015 IEEE International Conference on Digital Signal Processing (DSP)*. IEEE, 2015, pp. 1076–1080.
- [32] A. Majumdar, “Improving synthesis and analysis prior blind compressed sensing with low-rank constraints for dynamic mri reconstruction,” *Magnetic resonance imaging*, vol. 33, no. 1, pp. 174–179, 2015.

- [33] Z. Li and J. Tang, “Weakly supervised deep matrix factorization for social image understanding,” *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 276–288, 2016.
- [34] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, “A deep matrix factorization method for learning attribute representations,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 3, pp. 417–429, 2016.
- [35] D. A. Spielman, H. Wang, and J. Wright, “Exact recovery of sparsely-used dictionaries,” in *Conference on Learning Theory. JMLR Workshop and Conference Proceedings*, 2012, pp. 37–1.
- [36] S. Arora, A. Bhaskara, R. Ge, and T. Ma, “More algorithms for provable dictionary learning,” *arXiv preprint arXiv:1401.0579*, 2014.
- [37] T. Goldstein and S. Osher, “The split bregman method for  $l_1$ -regularized problems,” *SIAM journal on imaging sciences*, vol. 2, no. 2, pp. 323–343, 2009.
- [38] Z. Qin, D. Goldfarb, and S. Ma, “An alternating direction method for total variation denoising,” *Optimization Methods and Software*, vol. 30, no. 3, pp. 594–615, 2015.
- [39] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo, “Fast image recovery using variable splitting and constrained optimization,” *IEEE transactions on image processing*, vol. 19, no. 9, pp. 2345–2356, 2010.

- [40] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [41] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.
- [42] C. Bao, Y. Quan, and H. Ji, “A convergent incoherent dictionary learning algorithm for sparse coding,” in *European Conference on Computer Vision*. Springer, 2014, pp. 302–316.
- [43] J. Wang, J.-F. Cai, Y. Shi, and B. Yin, “Incoherent dictionary learning for sparse representation based image denoising,” in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 4582–4586.
- [44] K. Basu, V. Debusschere, S. Bacha, U. Maulik, and S. Bondyopadhyay, “Nonintrusive load monitoring: A temporal multilabel classification approach,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 262–270, 2014.
- [45] N. Batra, M. Gulati, A. Singh, and M. B. Srivastava, “It’s different: Insights into home energy consumption in india,” in *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, 2013, pp. 1–8.

- [46] T. Ganu, D. P. Seetharam, V. Arya, R. Kunnath, J. Hazra, S. A. Husain, L. C. De Silva, and S. Kalyanaraman, “nplug: A smart plug for alleviating peak loads,” in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, 2012, pp. 1–10.
- [47] S. Singh and A. Majumdar, “Deep sparse coding for non-intrusive load monitoring,” *IEEE Transactions on Smart Grid*, 2017.
- [48] S. Makonin, F. Popowich, I. V. Bajić, B. Gill, and L. Bartram, “Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring,” *IEEE Transactions on smart grid*, vol. 7, no. 6, pp. 2575–2585, 2015.
- [49] D. Erhan, A. Courville, Y. Bengio, and P. Vincent, “Why does unsupervised pre-training help deep learning?” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings*, 2010, pp. 201–208.
- [50] M. Zeifman and K. Roth, “Nonintrusive appliance load monitoring: Review and outlook,” *IEEE transactions on Consumer Electronics*, vol. 57, no. 1, pp. 76–84, 2011.
- [51] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar, “Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey,” *Sensors*, vol. 12, no. 12, pp. 16 838–16 866, 2012.

- [52] D. Li and S. Dick, “Residential household non-intrusive load monitoring via graph-based multi-label semi-supervised learning,” *IEEE Transactions on Smart Grid*, 2018.
- [53] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2008.
- [54] H. Larochelle and Y. Bengio, “Classification using discriminative restricted boltzmann machines,” in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 536–543.
- [55] P. Smolensky, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. SIAM, 1986.
- [56] G. Hinton, “Training products of experts by minimizing contrastive divergence,” vol. 14, August 2002, pp. 1771–1800.
- [57] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio, “Learning algorithms for the classification restricted boltzmann machine,” vol. 13, March 2012, pp. 643–669.
- [58] B. Marlin, K. Swersky, B. Chen, and N. Freitas, “Inductive principles for restricted boltzmann machine learning,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 509–516.

- [59] G. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” vol. 313. American Association for the Advancement of Science, 2006, pp. 504–507.
- [60] R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted boltzmann machines for collaborative filtering,” in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 791–798.
- [61] U. Fiore, F. Palmieri, A. Castiglione, and A. Santis, “Network anomaly detection with the restricted boltzmann machine,” vol. 122, 2013, pp. 13–23.
- [62] A. Coates, A. Ng, and H. Lee, “An analysis of single-layer networks in unsupervised feature learning,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 215–223.
- [63] X. Li, F. Zhao, and Y. Guo, “Conditional restricted boltzmann machines for multi-label learning with incomplete labels,” in *AISTATS*, 2015.
- [64] S. Verma, P. Patel, and A. Majumdar, “Collaborative filtering with label consistent restricted boltzmann machine,” in *Ninth International Conference on Advances in Pattern Recognition*, Dec. 2017.
- [65] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *Proc. of ICLR (2015)*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6980>

- [66] Y. Kongsorot and P. Horata, “Multi-label classification with extreme learning machine,” 2014.
- [67] E. Gomes and L. Pereira, “Pb-nilm: Pinball guided deep non-intrusive load monitoring,” *IEEE Access*, vol. 8, pp. 48 386–48 398, 2020.
- [68] G. Bejarano, D. DeFazio, and A. Ramesh, “Deep latent generative models for energy disaggregation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 850–857.
- [69] K. He, D. Jakovetic, B. Zhao, V. Stankovic, L. Stankovic, and S. Cheng, “A generic optimisation-based approach for improving non-intrusive load monitoring,” *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6472–6480, 2019.
- [70] B. Zhao, L. Stankovic, and V. Stankovic, “On a training-less solution for non-intrusive appliance load monitoring using graph signal processing,” *IEEE Access*, vol. 4, pp. 1784–1799, 2016.
- [71] L. Zhang, W.-D. Zhou, P.-C. Chang, J. Liu, Z. Yan, T. Wang, and F.-Z. Li, “Kernel sparse representation-based classifier,” *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1684–1695, 2011.
- [72] A. Majumdar and R. K. Ward, “Fast group sparse classification,” *Canadian Journal of Electrical and Computer Engineering*, vol. 34, no. 4, pp. 136–144, 2009.

- [73] T. Guha and R. K. Ward, “Learning sparse representations for human action recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 8, pp. 1576–1588, 2011.
- [74] E. J. Candes and T. Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?” *IEEE transactions on information theory*, vol. 52, no. 12, pp. 5406–5425, 2006.
- [75] E. J. Candès and M. B. Wakin, “An introduction to compressive sampling,” *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.
- [76] D. L. Donoho, “Compressed sensing: Ieee transactions on information theory,” 2006.
- [77] M. Z. A. Bhotto, S. Makonin, and I. V. Bajić, “Load disaggregation based on aided linear integer programming,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 7, pp. 792–796, 2016.
- [78] F. M. Wittmann, J. C. López, and M. J. Rider, “Nonintrusive load monitoring algorithm using mixed-integer linear programming,” *IEEE Transactions on Consumer Electronics*, vol. 64, no. 2, pp. 180–187, 2018.
- [79] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, “Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors,” *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.
- [80] J. Yoo, S. Becker, M. Monge, M. Loh, E. Candes, and A. Emami-Neyestanak, “Design and implementation of a fully integrated compressed-



- sensing signal acquisition system,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5325–5328.
- [81] S. Foucart, “A note on guaranteed sparse recovery via  $l_1$ -minimization,” *Applied and Computational Harmonic Analysis*, vol. 29, no. 1, pp. 97–103, 2010.
- [82] S. K. Sahoo and A. Makur, “Signal recovery from random measurements via extended orthogonal matching pursuit,” *IEEE Transactions on Signal Processing*, vol. 63, no. 10, pp. 2572–2581, 2015.
- [83] I. Tošić and P. Frossard, “Dictionary learning,” *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 27–38, 2011.
- [84] D. L. Donoho, “For most large underdetermined systems of linear equations the minimal  $l_1$ -norm solution is also the sparsest solution,” *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 59, no. 6, pp. 797–829, 2006.
- [85] E. J. Candes and Y. Plan, “A probabilistic and ripsless theory of compressed sensing,” *IEEE transactions on information theory*, vol. 57, no. 11, pp. 7235–7254, 2011.
- [86] M. Aharon, M. Elad, and A. Bruckstein, “*rmk*-svd: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

- [87] S. Gleichman and Y. C. Eldar, “Blind compressed sensing,” *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6958–6975, 2011.
- [88] A. Majumdar, *Compressed sensing for engineers*. CRC Press, 2018.
- [89] S. Singh, V. Singhal, and A. Majumdar, “Deep blind compressed sensing,” *arXiv preprint arXiv:1612.07453*, 2016.
- [90] S. Tariyal, A. Majumdar, R. Singh, and M. Vatsa, “Deep dictionary learning,” *IEEE Access*, vol. 4, pp. 10 096–10 109, 2016.
- [91] V. Singhal, A. Majumdar, and R. K. Ward, “Semi-supervised deep blind compressed sensing for analysis and reconstruction of biomedical signals from compressive measurements,” *IEEE Access*, vol. 6, pp. 545–553, 2017.
- [92] C. Dinesh, B. W. Nettasinghe, R. I. Godaliyadda, M. P. B. Ekanayake, J. Ekanayake, and J. V. Wijayakulasooriya, “Residential appliance identification based on spectral information of low frequency smart meter measurements,” *IEEE Transactions on smart grid*, vol. 7, no. 6, pp. 2781–2792, 2015.