Received May 14, 2018, accepted June 13, 2018, date of current version September 7, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2850707

Disaggregating Transform Learning for Non-Intrusive Load Monitoring

MEGHA GAUR[1] AND ANGSHUL MAJUMDAR[2], (Senior Member, IEEE)
[1]Department of Computer Science, Indraprastha Institute of Information Technology Delhi, New Delhi 110020, India
[2]Department of Electrical and Computer Engineering, Indraprastha Institute of Information Technology Delhi, New Delhi 110020, India

Corresponding author: Megha Gaur (meghag@iiitd.ac.in)

The work of M. Gaur was supported by the TCS Ph.D. Fellowship. The work of A. Majumdar was supported in part by the Infosys Center for Artificial Intelligence at IIIT Delhi and in part by the DST IC-IMPACTS Indo-Canadian Grant.

ABSTRACT This paper addresses the problem of energy disaggregation/non-intrusive load monitoring. It introduces a new method based on the transform learning formulation. Several recent techniques, such as discriminative sparse coding, powerlet disaggregation, and deep sparse coding, are based on the synthesis dictionary learning/sparse coding approach; ours is based on its analysis equivalent. The theoretical advantage of analysis dictionary compared to its synthesis counterpart is that the former can learn from fewer training samples—this has implications in reducing the cost of energy disaggregation. Experiments have been carried out on two benchmark data sets—REDD and Dataport (Pecan Street). Comparison has been done with factorial HMM, discriminative sparse coding, powerlet disaggregation, and deep sparse coding. In the low training data regime, our method always excels over the others.

INDEX TERMS Dictionary learning, energy disaggregation, non-intrusive load monitoring.

I. INTRODUCTION

Non intrusive load monitoring (NILM) is a single channel blind source separation problem, where the smart-meter is the only channel that reads the total power consumption, and the problem is to find the power consumption of individual appliances. This means that there is a single observation (smart meter reading) and many variables (appliances); this is a highly under-determined problem and hence impossible to solve directly.

Today, homes and offices (not accounting industrial loads) consume about 40% of the total energy; studies estimate that 20% of it could be saved by modification in user behavior [1]. Energy disaggregation via non-intrusive load monitoring estimates the consumption patterns of individuals. It is believed that by giving a feedback on the consumption pattern, the awareness of the consumer can be increased which in turn would reduce wastage [2].

The earliest work in NILM dates back to the early 90's [3]. In those days, most of the appliances were ON-OFF devices (e.g. fan, light, heater, oven, cooler etc.); therefore modelling them as finite state machines (FSM) was a reasonable idea. With time, more complex multi-state appliances were introduced, e.g. computers, printers, etc. These did not show quantum increase/decrease in power, consumption varied smoothly. FSM failed to model such appliances. More recent techniques, based on stochastic finite state machines (Factorial Hidden Markov Models) [4], [5], have improved upon the prior approach. Such techniques can handle noise in multi-state appliances but still are not optimal for continuously varying loads.

Dictionary learning and sparse coding [6], [7] is the most current approach in energy disaggregation. Such dictionary learning based methods are not limited by the assumptions of stochastic finite state machines and hence are capable of handling all kinds of loads – multi-state and continuously varying.

There is yet another class of methods that is gaining popularity in recent times based on the multi-label classification approach [8]–[10]. These do not model the electrical appliance in any way but want to predict the state of the appliance (ON/OFF) given the aggregate power signal. Since multiple appliances can be ON at the same time, this turns out to be a multi-label classification problem. However these techniques cannot estimate the consumption accurately.

The generality of dictionary learning methods in energy disaggregation motivates our work. Standard dictionary learning is a synthesis formulation. It learns a basis so as to synthesize the data along with the learnt coefficients. In the past decade, it has enjoyed significant amount of success in machine learning and signal processing. Our work is based on the analysis version of dictionary learning – Transform learning [11]–[13].

Most NILM techniques are based on a learning based paradigm. The training stage is intrusive. It requires instrumenting the homes at plug level for collecting data that is later used for learning appliance specific models. These models are later used for disaggregating during the operational phase; at this stage the plug level sensors are removed. There are a few recent studies that do not need actual power consumption from every device and can act on aggregate data; they are based on the multi-label classification paradigm [8]–[10]. Studies such as [14]–[16] proposes completely unsupervised techniques for energy disaggregation. However, to the best of our knowledge, the supervised techniques are significantly more accurate than these.

For supervised approaches, data acquisition at the training phase is expensive owing to the cost of buying/renting the plug level sensors – this determines the 'cost' of NILM. If one can reduce the training phase by reducing the period over which data needs to be collected (for training mode disaggregation) or reduce the number of homes from which the data needs to be collected (for testing mode) the cost of NILM will reduce proportionately. This will have significant cost implications on the utilities and the consumers. More number of consumers will benefit without increase in any data acquisition cost.

Everything else remaining the same, a transform can generalize better than a dictionary, i.e. it has better representation capability. In other words, for a problem of fixed complexity the size of the transform can be much smaller than a given dictionary. Therefore, given limited volume of training data, the smaller sized transform will be less prone to over-fitting than the dictionary. This is the major benefit of transform learning – it can learn from far less training data. This in turn means that practical advantage mentioned in the previous paragraph will become feasible. This motivates our proposed formulation.

Relevant literature will be reviewed in the next section. Our proposed formulation is given in section III. The experimental results are discussed in section 4. The concluding remarks are in section 5.

II. LITERATURE REVIEW

A. SYNTHESIS SPARSE CODING

The idea of learning a basis for modelling each appliance was introduced by Kolter *et al.* [6]. It follows the typical NILM scenario. Training data is collected over time, where the smart meter logs consumptions from every single device. This is achieved by plug level monitors (such as jPlug). The training data is expressed as X_i (real power) where i is the index for an appliance. For each appliance they learnt a dictionary, i.e. they expressed:

$$X_i = D_i Z_i, \quad i = 1 \ldots \mathrm{N} \tag{1}$$

where D_i represents the basis/dictionary, Z_i are the loading coefficients, assumed to be sparse and N is the total number of appliances. X_i will have a dimension of hourly sampling rate along the rows and dimension of 24 (hours) x the number of training days along the columns. The number of dictionary atoms (columns) for D_i has to be specified by the user.

In the training phase a dictionary is learnt to model each appliance. This is achieved by solving –

$$\min_{D_i, Z_i} \|X_i - D_i Z_i\|_F^2 + \lambda \|Z_i\|_1, \quad i = 1 \ldots \mathrm{N} \tag{2}$$

Since Z_i's are supposed to be loading coefficients, they are supposed to be positive. This is assured by projecting the solution of (2), from every iteration to the positive space.

The basic interpretation of the dictionary is that its columns act as an abstract basis for representing an appliance. The power consumption is therefore a linear combination of these basis. To give a more concrete example, consider an electric fan; the columns/basis in the dictionary can be thought of as its distinct states (say 1 to 5); the coefficients then are the proportions of how long each state had run during the time period. This is the reason for the positivity constraint.

This concludes the basic dictionary learning approach to NILM. In a quest to improve the results, [6] introduced other complicated penalties; however in practice the pay off from these penalties was nominal since the results did not improve much over the basic formulation.

This model (2) does not account for the time varying nature of the appliances. This was accounted for in [7] where they introduced an auto-regressive model on the dictionary atoms.

So far we have discussed about the training stage. The test/operational stage remains the same for all methods. Let X denote the total power from N appliances (the columns indicate smartmeter readings over the same period of time as in training). This is expressed as:

$$X = \sum_{i=1}^{N} X_i = \sum_{i=1}^{N} D_i Z_i \tag{3}$$

Given the additive model, one can estimate the loading coefficients for each appliance by solving the following sparse recovery problem,

$$\min_{Z_1,\ldots,Z_N} \left\| X - [D_1|\ldots|D_N] \begin{bmatrix} Z_1 \\ \ldots \\ Z_N \end{bmatrix} \right\|_F^2 + \lambda \left\| \begin{matrix} Z_1 \\ \ldots \\ Z_N \end{matrix} \right\|_1 \tag{4}$$

The interpretation here is that, given the basis for each device, one needs to estimate the corresponding loading coefficients.

As before, positivity constraints are enforced on the loading coefficients estimated from (4). The formulation for disaggregation (4) is convex. From the estimated loading coefficients the device level power consumption can be computed.

$$\hat{X}_i = D_i Z_i, \quad i = 1 \ldots \mathrm{N} \tag{5}$$

In a very recent work [17], a deep version of sparse coding has been proposed. In there, multiple layers of dictionaries are learnt for each appliance; the rest of the mechanism remains the same. This is by far the state-of-the-art for standard supervised evaluation protocols.

In a recent work [18], a co-sparse analysis formulation has been proposed. Co-sparsity means that the signal is sparse under analysis. The main difference from the sparse coding/dictionary learning formulation is that, [18] learns a co-sparsity promoting dictionary. This is given by

$$\min_{D_i,\hat{X}_i} \left\| X_i - \hat{X}_i \right\|_F^2 + \lambda \left\| D_i \hat{X}_i \right\|_1 \tag{6}$$

Here co-sparsity is accounted for by the $\left\| D_i \hat{X}_i \right\|_1$ term. A device specific analysis basis is learnt such that the clean version of the data ($\hat{X}_i$) is co-sparse. The major motivation for moving from the synthesis to the analysis paradigm is to reduce the problem of over-fitting in limited data regimes. This will be explained in detail later.

B. TRANSFORM LEARNING

Dictionary learning is a synthesis formulation (Fig. 1a), it synthesizes/generates the data (X) from the linear combination of atoms (D) and learnt coefficients (Z). It has been used profusely for analysis [18], [19] and synthesis problems [20]. Transform learning is the analysis equivalent (Fig. 1b). It learns a transform (T) so that it operates/analyses the data (X) to generate the coefficients (Z).

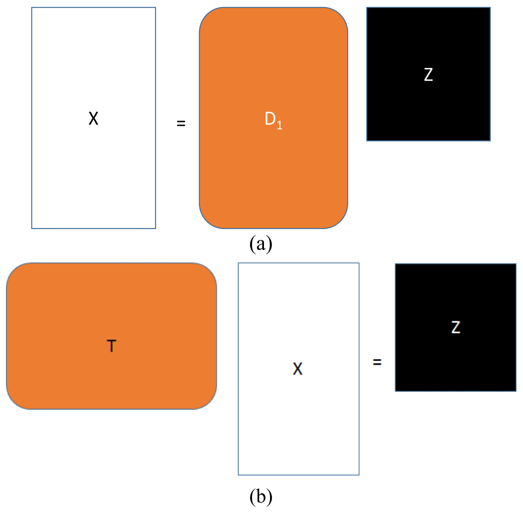


FIGURE 1. (a) Dictionary Learning; (b) Transform Learning.

Unlike dictionary learning/sparse coding, transform learning is relatively new. For the interested reader, we request to peruse [11], [12]. We will discuss the formulation briefly for the sake of completeness. Transform learning analyses the data by learning a transform/basis to produce coefficients. Mathematically this is expressed as,

$$TX = Z \tag{7}$$

Here T is the transform, X is the data and Z is the corresponding coefficients. The data matrix X consists of the features along the rows and samples along the columns. The number of transform atoms are determined by the user; this also defines the coefficient matrix Z.

The following transform learning formulation was proposed in [11] and [12] –

$$\min_{T,Z} \|TX - Z\|_F^2 + \lambda \left(\|T\|_F^2 - \log\det T \right) + \mu \|Z\|_1 \tag{8}$$

The factor $-\log\det T$ imposes a full rank on the learned transform; this prevents the degenerate solution ($T = 0$, $Z = 0$). The additional penalty $\|T\|_F^2$ is to balance scale; without this $-\log\det T$ can keep on increasing producing degenerate results in the other extreme.

An alternating direction approach was proposed to solve (8). This is given by –

$$Z \leftarrow \min_{Z} \|TX - Z\|_F^2 + \mu \|Z\|_1 \tag{9a}$$

$$T \leftarrow \min_{T} \|TX - Z\|_F^2 + \lambda \left(\varepsilon \|T\|_F^2 - \log\det T \right) \tag{9b}$$

Updating the coefficients (9a) is straightforward. It can be updated via one step of soft thresholding. This is expressed as,

$$Z \leftarrow signum(TX) \cdot \max(0, abs(TX) - \mu) \tag{10}$$

Here '·' indicates element-wise product.

In the initial paper on transform learning [6], a non-linear conjugate gradient based technique was proposed to solve the transform update. In the more refined version [7] with some linear algebraic tricks they were able to show that a closed form update exists for the transform.

$$XX^T + \lambda\varepsilon I = LL^T \text{(cholesky decomposition)} \tag{11a}$$

$$L^{-1}XZ^T = USV^T \text{(SVD)} \tag{11b}$$

$$T = 0.5V \left(S + (S^2 + 2\lambda I)^{1/2} \right) Q^T L^{-1} \tag{11c}$$

The first step is to compute the Cholesky decomposition; the decomposition exists since $XX^T + \lambda\varepsilon I$ is symmetric positive definite. The next step is to compute the full SVD. The final step is the update step. The proof for convergence of such an update algorithm can be found in [13].

There are only a handful of papers on this topic. Theoretical aspects of transform learning are discussed in the aforesaid papers on transform learning. So far it has limited visibility outside the signal processing community. The only application of transform learning in machine learning has been by [21] and [22]; where the same formulation has been dubbed as 'analysis sparse coding'. There it was used for simple unsupervised feature extraction. A later study [20] proposed a discriminative version for supervised feature extraction. In signal processing, it is mainly used for solving inverse problems [24]–[26].

III. PROPOSED FORMULATION

Today dictionary learning is a popular representation learning tool. Standard dictionary learning is a synthesis approach. However, recent studies in analysis dictionary learning (such as [18]) empirically show improvement over its traditional synthesis counterpart; especially in limited data regimes. Analysis dictionary learning/transform learning is less prone to over-fitting [18], [22], and [23], i.e. can learn from fewer samples.

In disaggregation, datasets such as REDD (testing mode) define 4:1 splits for training and testing [27]. These are impractical training scenarios. In most practical cases, it is not possible to instrument so many houses with sensors – the situation is exactly the opposite. One needs to disaggregate/test on a large number of houses by learning from far fewer labeled (instrumented) households.

For the training mode (where the data from the same house if used for training), one needs to ensure that the number of training days is minimized. This would directly benefit the utilities and consumers. It would enable the utilities to instrument more houses (for collecting training data) and thus bring the benefits of disaggregation to more consumers.

In such data scarce scenarios, it is likely that, analysis transform learning, with its capacity to learn from fewer samples will yield better generalizability on unseen (test) cases than the corresponding synthesis dictionary learning technique.

A. TRAINING

The methodology/protocol is exactly the same as in dictionary learning [6]. As before, given the training data for each device, we learn a device specific transform. Assuming X_i is the training data for the i^{th} device, this is expressed as,

$$T_i X_i = Z_i \tag{12}$$

The straightforward way to solve this problem is to learn one transform for each device in a naïve fashion. Here X_i has the same meaning as (1)

$$\min_{T_i, Z_i} \sum_i \|T_i X_i - Z_i\|_F^2 + \lambda \left(\|T_i\|_F^2 - \log\det T_i \right) + \mu \|Z_i\|_1 \tag{13}$$

For disaggregation we would expect that the transforms are discriminative, i.e. the transform for the i^{th} appliance should only produce sparse codes for the i^{th} appliance but not represent any other appliances; i.e. should not generate sparse codes for other appliance. This means that $T_i X_{i^C}$ (the superscript 'c' on 'i' indicates complimentary set of appliances) should not be sparse – they should be dense and small.

The basic formulation (15) does not enable this. Therefore we need to modify (15). We achieve this by adding the additional term (for each i) $\|T_i X_{i^C}\|_F^2$; the Frobenius norm would ensure that the coefficients obtained from $T_i X_{i^C}$ should be dense and very small (approximately Normal distribution). Incorporating these terms into (15) leads to,

$$\min_{T_i, Z_i} \sum_i \|T_i X_i - Z_i\|_F^2 + \lambda \left(\|T_i\|_F^2 - \log\det T_i \right) + \mu \|Z_i\|_1 + \gamma \|T_i X_{i^C}\|_F^2 \tag{14}$$

We follow the same alternating minimization approach as proposed by [11] and [12]. The formulation (16) can be decoupled for each device, leading to –

$$\min_{T_i, Z_i} \|T_i X_i - Z_i\|_F^2 + \lambda \left(\|T_i\|_F^2 - \log\det T_i \right) + \mu \|Z_i\|_1 + \gamma \|T_i X_{i^C}\|_F^2 \tag{15}$$

Alternating minimization leads to,

$$\min_{T_i} \|T_i X_i - Z_i\|_F^2 + \lambda \left(\|T_i\|_F^2 - \log\det T_i \right) + \gamma \|T_i X_{i^C}\|_F^2 \tag{16a}$$

$$\min_{Z_i} \|T_i X_i - Z_i\|_F^2 + \mu \|Z_i\|_1 \tag{16b}$$

The sparse coding step (16b) remains exactly the same as before (9b). Hence can be solved using (8).

For the transform update, we can express (16a) as,

$$\min_{T_i} \left\| T_i \left[X_i | \sqrt{\gamma} X_{i^C} \right] - [Z_i | 0] \right\|_F^2 + \lambda \left(\|T_i\|_F^2 - \log\det T_i \right) \tag{17}$$

The $[\cdot|\cdot]$ means that the matrices are stacked horizontally. This brings the transform update to the same form as (9a). Hence we can use the same technique as (11).

In a succinct fashion, the entire training algorithm can be expressed in Algorithm 1.

Algorithm 1 Transform Learning for Load Disaggregation

```
For every appliance $i$ solve:
    $\min_{T_i, Z_i} \|T_i X_i - Z_i\|_F^2 + \lambda \left( \|T_i\|_F^2 - \log\det T_i \right) + \mu \|Z_i\|_1 + \gamma \|T_i X_{i^C}\|_F^2$
Initialize: Compute SVD: $X_i = USV^T$; initialize $Z_i = SV^T$
Until convergence run
    $\left( X_i | \sqrt{\gamma} X_{i^C} \right) \begin{pmatrix} X_i^T \\ \sqrt{\gamma} (X_{i^C})^T \end{pmatrix} + \lambda \varepsilon I = LL^T$
    $L^{-1} \left( X_i | \sqrt{\gamma} X_{i^C} \right) \begin{pmatrix} Z_i^T \\ 0 \end{pmatrix} = USV^T$
    $T_i = 0.5R \left( S + (S^2 + 2\lambda I)^{1/2} \right) Q^T L^{-1}$
    $Z_i \leftarrow signum(T_i X_i) \cdot \max(0, abs(T_i X_i) - \mu)$
```

Note that even though both the proposed formulation and [18] are based on the analysis paradigm, they are completely different. In [18] an analysis basis is learnt so as to 'clean' the data; it does not learn any representation. In our proposed formulation, we learn an analysis transform to generate the coefficients.

B. TESTING

During testing, the transform based disaggregation is not as straightforward as the dictionary learning based formulation; it needs further analysis. We start with the standard model that the total power is the sum of the power consumed by the individual devices. This is expressed as,

$$X = \sum_i X_i \tag{18}$$

Applying the learnt transform leads to –

$$\begin{bmatrix} T_1 \\ \dots \\ T_N \end{bmatrix} (X_1 + X_2 + \dots + X_N) \tag{19}$$

$$= \begin{bmatrix} T_1 (X_1 + X_2 + \dots + X_N) \\ \dots \\ T_N (X_1 + X_2 + \dots + X_N) \end{bmatrix}$$
$$= \begin{bmatrix} T_1 X_1 \\ \dots \\ T_N X_N \end{bmatrix} + \begin{bmatrix} T_1 (X_2 + \dots + X_N) \\ \dots \\ T_N (X_1 + X_2 + \dots + X_{N-1}) \end{bmatrix} \tag{20}$$

The terms $T_i X_{i^C}$'s will be close to zero or negligibly small – this follows from our training formulation. We have learnt the transforms in such a manner that the transforms for one device when applied on the data for another, will produce almost zero valued coefficients. This allows representing (20) as,

$$= \begin{bmatrix} T_1 X_1 \\ \dots \\ T_N X_N \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_N \end{bmatrix} \tag{21}$$

Here $T_1 (X_2 + \dots + X_N) = \varepsilon_1$, $T_2 (X_1 + X_3 \dots + X_N) = \varepsilon_2$ and so on. The error terms ε_i's are small (approximately Normal distribution). In a concise fashion, from (18), (20) and (21) we have the following expression,

$$\begin{bmatrix} T_1 \\ \dots \\ T_N \end{bmatrix} X = \begin{bmatrix} T_1 X_1 \\ \dots \\ T_N X_N \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_N \end{bmatrix} = \begin{bmatrix} Z_1 \\ \dots \\ Z_N \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_N \end{bmatrix} \tag{22}$$

where $T_i X_i = Z_i$ as per definition of the transform.

The transforms for each device have already been learnt during the training phase. Therefore one can solve for the coefficients from (22) by sparse coding.

$$\min_{Z_i' s} \left\| \begin{bmatrix} T_1 \\ \dots \\ T_N \end{bmatrix} X - \begin{bmatrix} Z_1 \\ \dots \\ Z_N \end{bmatrix} \right\|_F^2 + \mu \left\| \begin{bmatrix} Z_1 \\ \dots \\ Z_N \end{bmatrix} \right\|_1 \tag{23}$$

The l_2-norm on the data-fidelity arises from the fact that the errors ε_i's are Normally distributed (by definition from the training phase).

Solving (23) is straightforward. It requires one step of soft-thresholding, similar to (10). Once the sparse codes are obtained, one needs to solve the following set of inverse problems to generate the corresponding power consumptions for each device.

$$T_i X_i = Z_i \tag{24}$$

This has an analytic solution – Moore Penrose pseudo-inverse.

$$X_i = \left(T_i^T T_i \right)^{-1} T_i^T Z_i \tag{25}$$

TABLE 1. Description of appliances in REDD.

House	Appliances
1	Electronics, Lighting, Refrigerator, Disposal, Dishwasher, Furnace, Washer Dryer, Smoke Alarms, Bathroom GFI, Kitchen Outlets, Microwave
2	Lighting, Refrigerator, Dishwasher, Washer Dryer, Bathroom GFI, Kitchen Outlets, Oven, Microwave, Electric Heat, Stove
3	Electronics, Lighting, Refrigerator, Disposal, Dishwasher, Furnace, Washer Dryer, Bathroom GFI, Kitchen Outlets, Microwave, Electric Heat, Outdoor Outlets
4	Lighting, Dishwasher, Furnace, Washer Dryer, Smoke Alarms, Bathroom GFI, Kitchen Outlets, Stove, Disposal, Air Conditioning
6	Lighting, Refrigerator, Disposal, Dishwasher, Washer Dryer, Kitchen Outlets, Microwave, Stove

Notice that even though the analysis of the Transform based testing algorithm is slightly more involved than the dictionary counterpart; operationally/computationally it is much simpler and efficient. Both (23) and (25) have closed form solutions. For (23) one only requires a simple thresholding step; for (25) one needs a matrix vector product (the pseudoinverses for the T_i's can be pre-computed). Thus, operationally it is very fast – capable of real-time disaggregation.

IV. RESULTS

Our proposed algorithm is tested on benchmark datasets. For the sake of reproducibility we experiment on two publicly available ones – REDD and Pecan Street.

In principle, we could have applied this technique on any kind of electrical signal. However, in practice, smart meter readings are more practical. Hence, we stick to the traditional datasets.

A. REDD DATASET

The REDD dataset [27] is of moderate size. It comprises of power consumptions from six different houses. Table 1 shows appliances in different houses. For each house, the total/aggregate electricity consumption as well as individual consumptions of about twenty different household devices are recorded. The data is available for a period of two weeks with a very high frequency sampling rate of 15kHz. Such a high sampling rate is impractical; to emulate real-life scenario the training and testing data is aggregated over a time period of 10 minutes. We follow the standard evaluation protocol where the 5th house is omitted owing to very limited availability of samples.

The disaggregation accuracy is defined by [27] as follows,

$$Acc = 1 - \frac{\sum_t \sum_n \left| \hat{x}_t^{(i)} - x_t^{(i)} \right|}{2 \sum_t \bar{x}_t}$$

where t denotes time instant and n denotes a device; the 2 factor in the denominator is to discount the fact that the absolute value will "double count" errors. Here y_t denotes the actual (measured) power, $\hat{y}_t$ the estimated power and $\bar{y}_t$ the mean of the actuals.

We benchmark our proposed technique against – the Factorial HMM (FHMM) [4], Powerlet based Energy Disaggregation (PED) [7], discriminating sparse coding (discSC) [6] and deep sparse coding (DSC) [15]; on the standard training protocols (with enough training data) DSC is the most accurate disaggregating technique known.

Note that in this work, we do not compare with multi-label classification techniques. The reason has been discussed at the onset. These methods can only estimate the state of the appliance and cannot estimate its consumption directly. Power consumption can be indirectly estimated but that gives a very crude value, which is worse than simple disaggregation techniques like FHMM.

There are two protocols for evaluation [27]. 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 protocol. In the second mode, the data from some households are used for training and the remaining ones are used for prediction (called 'testing').

Usually the split between training and testing is 4:1. This is an overtly optimistic scenario. In real life situations, the training data will always be small compared to the testing data. In this work, we will show that for smaller volumes of testing data (practical situation), our proposed method yields better results than all other techniques.

We first show results for the training mode. The results are shown in Fig. 2. We are showing the mean disaggregation accuracy over all houses. In the graph, the X-axis shows the percentage of data (from each house) used for training; the Y-axis the corresponding aggregate (over all houses) disaggregation accuracy on the remaining portion (of each house). Since this is a small dataset, in order to make the results reproducible we have always taken the portion of training data starting from the beginning.

The parameter setting for the benchmarking techniques have been taken from the respective studies. Even though, the training and testing ratios differ between the aforesaid work and ours, the configurations will remain the same. This is because the configuration, for example the number of states of FHMM or the number of basis used for discSC, PED or DSC are representatives of the complexity of the device and not the volume of training data available. Since the devices remain the same, the configurations remain the same as well.

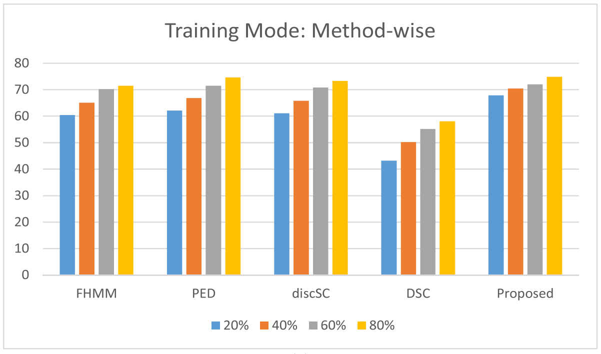
For FHMM, we use the function from NILMTK[1]; it is available there with pre-tuned parameters for these datasets. For discSC, we need to specify the sparsity promoting parameter (which is 2), the discriminative parameter (which is 0.001) and the number of atoms (twice redundant). For DSC, we use the sparsity promoting parametric value of 0.1 and the number of dictionary atoms are reduced by half in each subsequent layer. For PED, we use the sparsity promoting prior and the co-occurrence parameters are both unity and the temporal smoothness parameter is 30. As mentioned before, all these values are obtained from the corresponding papers.

For our proposed method, we have used 6 transform basis for each device; this is not a true representative of the complexity, e.g. a laptop or washing machine is more complex than a CFL or stove, and hence would require more basis – but such device specific fine tuning is time consuming. The parametric values used for transform learning have been tuned using the greedy L-curve method [28]. Here we first put μ to zero and tune λ; we obtain $\lambda = 0.1$ by the L-curve method. Then we fix the value of λ and tune μ to get a value $\mu = 0.5$.

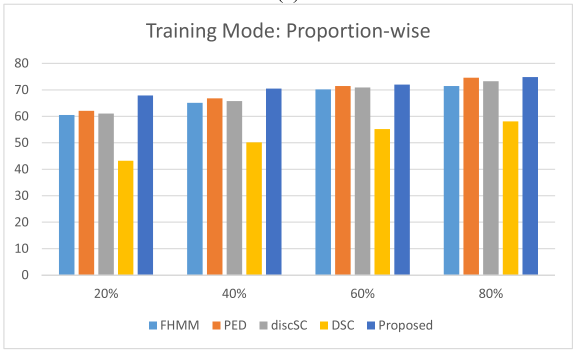
In Fig. 2a we show how different methods perform with change in training volume; the results are grouped by the disaggregation technique. In Fig. 2b we show the same results in a different fashion; we see how different methods perform for a fixed training volume.

Especially from Fig. 2a, we find that our proposed method is the most robust. There is only a small drop in disaggregation accuracy across the various proportions of training data; the drop is around 7%. All other dictionary learning/sparse coding based methods drop more than 12%.

In [15] it has been claimed that deep sparse coding yields the best results; but we can see here that for limited training



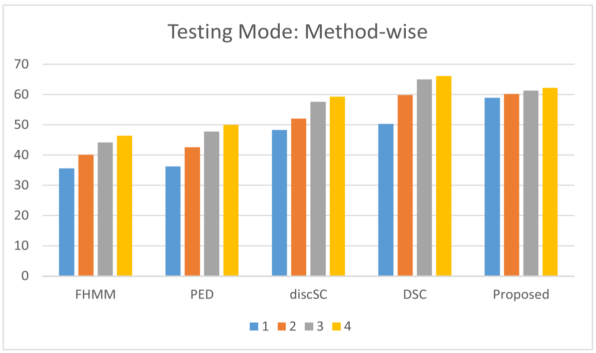
(a)



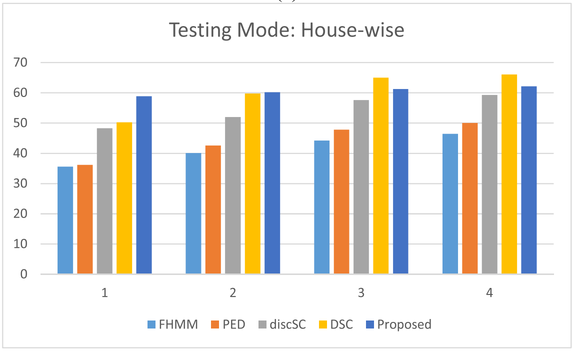
(b)

FIGURE 2. **REDD Training Mode Disaggregation Results. Y-axis shows the disaggregation accuracy.**

[1]https://github.com/nilmtk

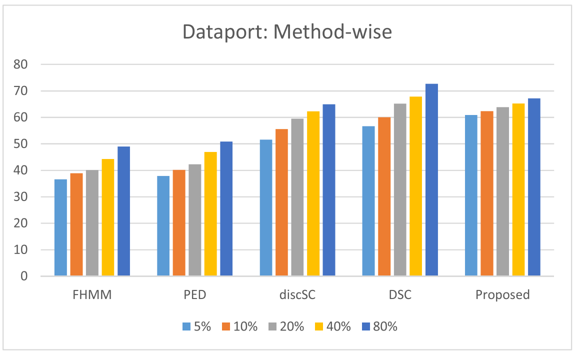


(a)

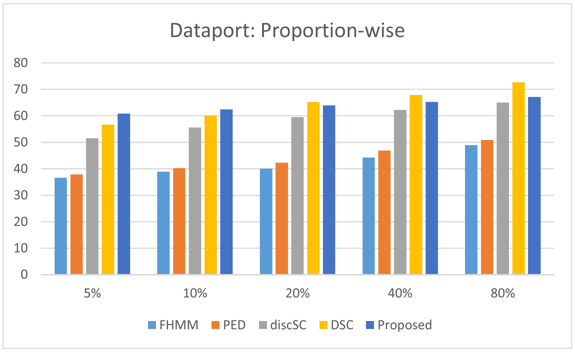


(b)

FIGURE 3. **REDD Testing Mode Disaggregation Results. Y-axis shows the disaggregation accuracy.**



(a)



(b)

FIGURE 4. **Dataport Testing Mode Disaggregation Results. Y-axis shows the disaggregation accuracy.**

volume it performs bad. This is true for deep learning in general; they only yield good results when the volume of training data is large.

The next set of experiments are in testing mode. We choose k houses; for each k, there are ^5C_k possible combinations of training houses. We carry out experiments on all of them and test on the remaining houses for each training set. The results are shown in Fig. 3. The average disaggregation accuracy is reported. As before we have shown the same result in two different ways. In Fig. 3a the variation within a technique for changing training volume (number of houses) is shown, and in Fig. 3b, the variation among the methods for a fixed training volume is shown.

One finds that the testing mode results show a similar trend; especially from Fig. 3a. As the number of houses (training data) decreases, the disaggregation accuracy suffers as well. But our method suffers the least drop in accuracy (less than 3% drop); all other methods show significant drops. FHMM and discSC drops by 12%, PED and DSC drops more than 15%. When the volume of training data is high, DSC yields better results than ours. But as the training volume decreases (practical scenario), we perform better.

B. DATAPORT PECAN STREET DATASET

The Dataport dataset is available in NILMTK (non-intrusive load monitoring toolkit) [29] format. Experiments were carried out on a subset of it. This dataset contains 1 minute circuit level and building level measurements from 240 houses. The readings have been collected from 18 different devices: air conditioner, kitchen appliances, electric vehicle, and electric hot tub heater, electric water heating appliance, dishwasher, spin dryer, freezer, furnace, microwave, oven, electric pool heater, refrigerator, sockets, electric stove, waste disposal unit, security alarm and washer dryer.

On this dataset, the usual protocol is to test on the 'testing' mode. Usually about 66% of the homes form the training dataset and the remaining 34% of the homes form the test set [29]. In this work, we train on fewer homes and test on the remaining; this is a more practical scenario. For each proportion of the training set, the homes are chosen randomly. This is done 100 times for each configuration. For each such configuration, the remaining homes are used for testing. The mean disaggregation accuracy from various techniques is shown in Fig. 3.

As we did before for REDD, the training and testing data are prepared by aggregating the data for a period of 10 minutes. This reflects real life scenario and is the usual protocol on this dataset.

As in the previous sub-section, the configuration for the benchmarking techniques have been taken from the respective papers; the same parametric values mentioned before have been used. The configuration for our proposed algorithm and parametric values also remain the same.

In Fig. 4a we show the variation of disaggregation accuracy within a technique for changing proportions of training data. In Fig. 4b, the same results are shown in a different fashion;

TABLE 2. Normalized error for common devices at 40% and 5% training volume.

Appliance	FHMM		discSC		PED		DSC		Proposed	
	40%	5%	40%	5%	40%	5%	40%	5%	40%	5%
AC	3.16	10.21	0.70	2.41	2.52	4.17	0.89	3.11	0.92	1.11
Dryer	51.47	87.03	2.04	9.12	35.69	41.12	1.11	6.20	1.24	1.57
Dishwasher	6.48	12.11	1.25	9.70	6.08	14.89	0.66	3.97	0.76	2.24
Microwave	4.96	10.72	0.84	2.17	4.34	9.48	0.76	1.26	0.75	2.15
Furnace	0.89	2.31	0.63	1.29	0.93	2.05	0.58	1.04	0.61	1.16
Fridge	2722.82	2900.51	516.31	847.56	986.30	1182.41	460.56	774.61	467.78	514.08
Washer	21.80	27.63	0.93	1.97	19.62	28.17	0.59	2.19	0.62	1.76

it shows the variation in accuracy among the methods given the training volume.

The results show a similar to that of REDD. As can be seen from Fig. 4a. For high volume of training data, deep sparse coding (DSC) yields very good results. But when the training volume decreases, the accuracy falls significantly; the drop in disaggregation accuracy is more than 15%. For FHMM and PED the drop in accuracy is around 13% and for discriminative sparse coding it is around 15%. For our proposed method, the drop is only 7%.

For this dataset, we show the normalized absolute error (defined below) on some common devices (indexed as i) in Table 2; the results are shown for two training volumes – 40% and 5%.

$$Err(i) = \frac{\sum_t \left|\hat{x}_t^{(i)} - x_t^{(i)}\right|}{\sum_t x_t^{(i)}}$$

The error metric normalizes the sum of the absolute differences between the predicted and actual power consumptions of the i^{th} device, summed over all instances of time.

One can see that as the training volume is reduced, the error increases sharply for all the method compared against; apart from the refrigerator and the washer, the errors for the other devices increase by two fold. Our method is the least perturbed by the extreme change in training volume. In the low training sample regime, our method yields the best results.

This has implications in cost; the training phase of energy disaggregation is intrusive; the appliances need to be sensed separately. Such sensors (such as jPlug) are expensive. By our method it will be possible to get the same disaggregation results with far lesser cost (fewer houses need to be instrumented for training mode and in testing mode fewer number of training days).

C. COMPARISON OF RUN-TIME

Consider the computational complexity of testing during sparse coding (and its variants). The expression is given in (4), repeated here for the sake of convenience.

$$\min_{Z_1,\ldots,Z_N} \left\| X - [D_1|\ldots|D_N] \begin{bmatrix} Z_1 \\ \ldots \\ Z_N \end{bmatrix} \right\|_F^2 + \lambda \left\| \begin{matrix} Z_1 \\ \ldots \\ Z_N \end{matrix} \right\|_1$$

This is an l_1-norm minimization problem. This needs to be solved iteratively and the usual complexity being a perturbed linear programming problem is $O(n^3)$. The same time complexity applies for other discSC, PED and DSC.

Our proposed method on the other hand requires solving (23) and (25), repeated here for the sake of convenience.

$$T_i X_i = Z_i$$
$$X_i = \left(T_i^T T_i\right)^{-1} T_i^T Z_i$$

Both of them are simple matrix products (the pseudo-inverse can be pre-computed since it only depends on the transform learnt during the training stage). The complexity of this is $O(n^2)$ – in optimization it is usually not possible to solve a problem any more efficiently. Thus, in theory our method is significantly faster than the sparse coding based techniques.

In terms of computational complexity during training, we need to solve two problems iteratively – (16a) and (16b). The cost of solving the sparse coding problem is $O(n^2)$. The cost of solving the transform update is $O(n^3)$ since it is dominated by the singular value decomposition. This cost is at par with the cost of dictionary learning. In there the complexity of updating the dictionary is $O(n^2)$ since it has a closed form update via the pseudoinverse. The cost of updating the sparse codes is $O(n^3)$ since it needs to be solved iteratively via l_1-minimization. The variants discSC and PED have the same complexity. The cost for solving the DSC problem (during training) increases linearly in the number of layers.

The experiments have been carried out on a desktop PC running 64 bit Windows 10. It has an i7 CPU clocked at 3.1 GHz. The RAM size is 16 GB. The testing and training run-times are shown in Tables 3 and 4, respectively. These results are shown for the usual protocols, i.e. 4:1 testing mode for REDD and 3:1 for Pecan Street.

TABLE 3. Testing times in seconds.

Method	REDD	Pecan Street
FHMM	3.1	50.3
discSC	3.5	125.6
PED	4.2	198.4
DSC	2.8	100.9
Proposed	0.1	4.7

The results corroborate the theory. Note that discSC is slightly slower than DSC because the overall dictionary size (after training) for deep sparse coding is smaller than that of discriminative sparse coding. Our proposed method is

TABLE 4. Training times in seconds.

Method	Pecan Street	REDD
FHMM	39.7	30.2
discSC	25.1	14.8
PED	29.6	17.7
DSC	43.9	25.3
Proposed	20.4	12.6

about an order of magnitude faster than sparse coding based methods; this is expected from theory.

In terms of training times, we see that the proposed method is comparable to other techniques. Is it slightly faster than the synthesis sparse coding techniques (although they have the same computational complexity) because it converges faster.

V. CONCLUSION

This paper proposes a new energy disaggregation/NILM technique. Our formulation is based on the transform (analysis equivalent of dictionary) learning formulation. The main advantage of the analysis formulation (as opposed to the synthesis sparse coding/dictionary learning) is that it has been seen to be less prone to over-fitting and hence can learn from fewer samples. We tested this capability experimentally in this paper; it has been seen that when the training volume is limited (practical scenarios) our method outperforms the state-of-the-art. With only 20% training data, our method can supersede results from standard approaches like FHMM (trained on 60% training data) and state-of-the-art approaches like deep sparse coding (trained on 80% training data).

This has implications on cost. The proposed technique can potentially reduce cost of instrumentation during the training phase. This allows the utilities to bring more customers under the umbrella in NILM.

The other additional advantage of analysis transform learning is operational speed. Since we are working on 10 min interval meter readings, this may not be of much importance right now – all techniques can disaggregate in this large time interval. But for faster sampling, the operational speed would become important. In such a scenario, our method would excel over others.

The proposed method excels over others at low training volumes. The advantages have been clearly pointed out. However, in cases where the training data is large, it loses out to deep techniques. This is seen across all domains of applied machine learning. We believe the only way to improve results for larger volumes is to propose a deeper architecture based on transform learning formulation. Some initial work on this topic has been done [30].

REFERENCES

[1] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel, "Disaggregated end-use energy sensing for the smart grid," *IEEE Pervasive Comput.*, vol. 10, no. 1, pp. 28–39, Mar. 2011.

[2] T. Babaei, H. Abdi, C. P. Lim, and S. Nahavandi, "A study and a directory of energy consumption data sets of buildings," *Energy Buildings*, vol. 94, pp. 91–99, May 2015.

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

[4] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial HMMs with application to energy disaggregation," *Artif. Intell. Statist.*, vol. 22, pp. 1472–1482, Mar. 2012.

[5] S. Makonin, F. Popowich, I. V. Bajić, B. Gill, and L. Bartram, "Exploiting HMM sparsity to perform online real-time nonintrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2575–2585, Nov. 2016.

[6] J. Z. Kolter, S. Batra, and A. Y. Ng, "Energy disaggregation via discriminative sparse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 1153–1161.

[7] E. Elhamifar and S. Sastry, "Energy disaggregation via learning powerlets and sparse coding," in *Proc. AAAI Conf. Artif. Intell.*, 2015, pp. 629–635.

[8] K. Basu, V. Debusschere, S. Bacha, U. Maulik, and S. Bondyopadhyay, "Nonintrusive load monitoring: A temporal multilabel classification approach," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 262–270, Feb. 2015.

[9] D. Li and S. Dick, "Whole-house non-intrusive appliance load monitoring via multi-label classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 2749–2755.

[10] S. M. Tabatabaei, S. Dick, and W. Xu, "Toward non-intrusive load monitoring via multi-label classification," *IEEE Trans. Smart Grid*, vol. 8, no. 1, pp. 26–40, Jan. 2017.

[11] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1072–1086, Mar. 2013.

[12] S. Ravishankar, B. Wen, and Y. Bresler, "Online sparsifying transform learning—Part I: Algorithms," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 625–636, Jun. 2015.

[13] S. Ravishankar and Y. Bresler, "Online sparsifying transform learning—Part II: Convergence analysis," *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 637–646, Jun. 2015.

[14] 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.

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

[16] R. Jia, Y. Gao, and C. J. Spanos, "A fully unsupervised non-intrusive load monitoring framework," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Nov. 2015, pp. 872–878.

[17] S. Singh and A. Majumdar, "Deep sparse coding for non-intrusive load monitoring," *IEEE Trans. Smart Grid*, Aug. 2017.

[18] S. Singh and A. Majumdar, "Analysis co-sparse coding for energy disaggregation," *IEEE Trans. Smart Grid*, Feb. 2017.

[19] Y. Xu, Z. Li, J. Yang, and D. Zhang, "A survey of dictionary learning algorithms for face recognition," *IEEE Access*, vol. 5, pp. 8502–8514, 2017.

[20] L. Jia *et al.*, "Image denoising via sparse representation over grouped dictionaries with adaptive atom size," *IEEE Access*, vol. 5, pp. 22514–22529, 2017.

[21] S. Shekhar, V. M. Patel, and R. Chellappa, "Analysis sparse coding models for image-based classification," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2014, pp. 5207–5211.

[22] J. Maggu and A. Majumdar, "Robust transform learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Mar. 2017, pp. 1467–1471.

[23] J. Guo, Y. Guo, X. Kong, M. Zhang, and R. He, "Discriminative analysis dictionary learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1617–1623.

[24] L. Pfister and Y. Bresler, "Tomographic reconstruction with adaptive sparsifying transforms," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2014, pp. 6914–6918.

[25] B. Wen, S. Ravishankar, and Y. Bresler, "Video denoising by online 3D sparsifying transform learning," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2015, pp. 118–122.

[26] S. Ravishankar and Y. Bresler, "Efficient blind compressed sensing using sparsifying transforms with convergence guarantees and application to magnetic resonance imaging," *SIAM J. Imag. Sci.*, vol. 8, no. 4, pp. 2519–2557, 2015.

[27] J. Z. Kolter and M. J. Johnson, "REDD: A public data set for energy disaggregation research," in *Proc. Workshop Data Mining Appl. Sustainability (SIGKDD)*, 2011, pp. 59–62.

[28] P. C. Hansen and D. P. O'Leary, "The use of the L-curve in the regularization of discrete ill-posed problems," *SIAM J. Sci. Comput.*, vol. 14, no. 6, pp. 1487–1503, 1993.

[29] N. Batra *et al.*, "NILMTK: An open source toolkit for non-intrusive load monitoring," in *Proc. 5th Int. Conf. Future Energy Syst. (e-Energy)*, 2014, pp. 265–276. [Online]. Available: https://github.com/nilmtk/nilmtk/wiki

[30] J. Maggu and A. Majumdar, "Greedy deep transform learning," in *Proc. IEEE ICIP*, Sep. 2017, pp. 1822–1826.

MEGHA GAUR received the master's degree from the Indraprastha Institute of Information Technology Delhi, New Delhi, in 2013, where she is currently pursuing the Ph.D. degree with the Department of Computer Science. Her research interests are in the areas of machine learning, signal processing, and their applications for energy conservation.

ANGSHUL MAJUMDAR received the bachelor's degree from the Bengal Engineering College, Shibpur, and the master's and Ph.D. degrees from The University of British Columbia in 2009 and 2012, respectively. He is currently an Assistant Professor with the Indraprastha Institute of Information Technology Delhi, New Delhi. He has co-authored over 150 papers in journals and reputed conferences. He has authored *Compressed Sensing for Magnetic Resonance Image Reconstruction* (Cambridge University Press). He is the Co-Editor of the *MRI: Physics, Reconstruction and Analysis* (CRC Press). His research interests are broadly in the areas of signal processing and machine learning. He is currently serving as the Chair of the IEEE SPS Chapter's Committee and the IEEE SPS Delhi Chapter.