New NILM methods for low-frequency smart meter data

SURYA VENKATESH PANDIYAN
venkatsurya460@gmail.com
Instituto Superior Tecnico, Universidade de Lisboa, Portugal

December 2020

Abstract

In this work, we proposed two new models variational autoencoder (VAE) based classifier and statistical (stats) rules-based temporal classifier for very low-frequency unsupervised Non-intrusive Load monitoring (NILM) and tested the state of art NILM method (Hidden Markov Models (HMM) with general priors) proposed for 10s data interval for 10min data interval. We adopted the HMM model for Factorial Hidden Markov Model (FHMM) to disaggregate the set of appliances from total consumption. We also proposed and tested a novel post-processing method based on K-Nearest Neighbour (K-NN), bottom-up approach, and similar time window (STW) methods. Our model verification shows that these three models are good enough only for the appliances water heater and space heater for real-world deployment. Our two methods VAE and statistical rules-based methods show superior performance than and similar performance to the state of art HMM method. Our detailed analysis shows that some of the issues of all three models can be solved by having already proposed sophisticated noise filters, data preparation, dependent appliance models, and post-processing. Our novel post-processing has proven to improve the Recall (Rec) value and F1-score even when its value is less than 0.1. this method can be integrated with any probabilistic NILM models. Our model validation shows that for very low frequency, the model cannot be completely non-intrusive and as well as high performing. Our results indicate NILM needs technological innovation and business innovation to gain trust and penetrate the market.

Keywords: Non-Intrusive Load Monitoring, smart grid, unsupervised learning, Hidden Markov Model, Variational Autoencoder, temporal classifier, K-Nearest Neighbor, bottom-up approach, similar time window, post-processing.

1.Introduction

The increase in global energy demand is a pressing concern for governments and utilities. The renewable energy to support these demands are intermittent in nature. It is, therefore, necessary to find methods by which we can tackle such demands without negative effects.

According to the international energy agency, the second major contributor to total electricity demand is the residential sector (23% in 2018 and 26% by 2040). So, it is important to optimize house energy demand/usage for cost reduction or energy reduction, or both. In this work, we focus on one such method-load deferral during peak demand to optimize house energy demand for cost reduction. Consumers would benefit from low peak prices and producers will benefit from unneeded additional investment for increasing production capacity or energy storage capacity to meet the peak demands. Load deferral is the concept of rescheduling appliances from peak demand period to low demand period.

To implement this technique, there is a need to know appliance power consumption in real-time. Deploying a meter for each appliance is undesirable due to higher manufacturing and installation cost, inconvenience, and maintenance. High-frequency total consumption readings provide more electrical features that help to find appliance consumption easily. However, this method is also undesirable because of higher manufacturing costs, and replacing already deployed low-frequency smart meter is a pain for electricity companies. The most promising economically viable option is to use the already deployed low-frequency smart meters. These smart meters are designed for billing. The consumption interval reading greater than one minute is not ideal for low power /multi-state devices disaggregation purposes. It contains very less useful features for disaggregation.

Signal processing/event-based Methods accuracy that only relies on total consumption would be low due to fewer features. Methods that rely on past appliance data is discouraging due to cost. In recent years, use the data collected from other houses to disaggregate for the house of interest is gaining interest due to its balance between cost and accuracy than the other two methods. This work is focusing on such methods. To summarize, the solution requires the following traits:

- Appliance load monitor: The method should estimate the appliance power consumption
- Non-intrusive: The method should use only a single point of data collection for all appliances in the house and collect minimum information from household members.
- Low-frequency data: The method should be suitable for appliance monitoring given aggregate consumption data at 5-15 minutes interval.
- Unsupervised: The method should not require training data for appliance monitoring from the house of interest.

1.1 Objective of the study

With the above solution requirements and challenges in mind, the proposed research objective for this thesis is as follows: To examine the performance of Popular unsupervised NILM methods proposed for low-frequency data(1s-1min) but not tested for very low-frequency data (5min-30 min). Test the practicality of these methods with less intrusion and/or survey. Develop new methods or adapt the existing methods for very low-frequency data to improve performance. Study the practicality of implementing peak demand response and find suitable appliances for it.

1.2 Background

Appliances signatures are the key to detect and classify appliances. Real power and reactive power are the most used steady-state features in very low-frequency algorithms. In this work, we used only real power.

The next step after feature extraction is load identification. Supervised learning is the most researched method in NILM. Hidden Markov Model (HMM) and neural networks (NN) are popular choices due to their ability to temporal data and state change information. For NILM various NN architectures have been tested, Multilayer perceptron, Convolutional Neural Network (CNN), long short-term memory (LSTM), Recurrent neural network (RNN), denoising Auto-encoders.

In unsupervised learning, the most researched model is HMM. This is due to the non-requirement of event detection. This is suitable for low-frequency samples as event detection is very difficult or not possible. In NILM, the Factorial Hidden Markov Model is popular where observation is the sum of the output from each Markov chain.

After an extensive literature review, considering multiple approaches have been taken and also many low-frequency algorithms are for sampling period between (1s-1min), we narrowed it down to four methods that are popular, capable to adapt to very low-frequency samplings, and suitable to the requirements we listed above.

In the study [1] generic model of appliances was used along with Difference-HMM (DHMM) to do load disaggregation. The generic model was tuned for every unseen house using aggregate signal alone.

Next to HMM, Neural network (NN)-based approaches are one of the popular methods for NILM. Usually, neural network-based methods are used in a supervised setting but also some of them tested their model to the houses not seen beforehand [2] some of them explored the possibilities and capabilities of neural networks for transfer learning [3] in the NILM context. We were surprised to find that Variational Auto encoder-based classification was not tested for NILM as far as we know. Variational Autoencoder based classification [4] is quite a popular technique for classification due to its ability to learn features as it is forced to compress data without losing its quality and also have been proposed for source separation problems [5]. We adopted this autoencoder based classification and proposed a new procedure for NILM.

The approach based on propositional concept learning is applied to NILM in [6]. It is a supervised technique. The step involves extracting sub-events from the training instances and by feature construction, the problem is represented in a way with these features for subsequent learning. These features represent temporal variation in the instance

Realizing the lack of efforts for postprocessing compared to disaggregation algorithm. We are proposing a K-NN and bottom-up approach-based classification technique as a possible post-processing tool for NILM.

A comprehensive review of the metrics used in NILM can be found in [7]. The metrics used in this thesis are Precision (Pre= TP/(TP+FP), recall (Rec= TP/(TP+FN), F-1 score(F1=(2xPxR)/(P+R). Our motive was only to find the on/off state of the appliances, so we considered the output as true positive(TP) and the appliance is on if both algorithm output and true value are above a small certain threshold say 10Wh. Similarly, we calculated True Negative (TN), False Positive (FP), False Negative (FN) to get the performance metrics.

The dataset used in this thesis is a subset called Irise from the European project residential monitoring to reduce energy use and carbon emissions in Europe (REMODECE)[8]. The Irise dataset consists of energy consumption readings of 100 houses for a whole year in France. The sampling time is 10 minutes. Three houses were used for training and another three houses were used for testing. The chosen appliances are a water heater, electric heating, washing machine, clothes drier, dishwasher.

2.Methodology

2.1 Model characteristics

In NILM, given the total electrical load, the goal is to find the individual consumption of appliances of interest. The models explored in this work are probabilistic models which provide the probability of the sequence which can be from the appliance of interest. Each appliance model is independent which means a model is created for each appliance and a single total consumption window can be accepted by multiple appliances though it violates the rule. During training, the model uses the train houses appliance's

data and during testing, the input to the model is the total consumption of the test houses. Any parameters that are required for eg: threshold probability to accept a window of data are decided by first testing the model for train houses with total consumption as input and select the one that provides maximum performance. For test houses, the methods are completely unsupervised. We also used Time Of Use (TOU) data from training data to know its assistance.

2.2 Hidden Markov Model

HMM is a variant of the Markov chain model in which the discrete state is not directly observed but inferred through continuous variable it emits. A continuous variable is a probabilistic distribution function of the state. Hidden states are represented as $Z=(z_1, z_2,\ldots\ldots z_T)$ $z\epsilon(1,\ldots K)$, where K is the number of discrete states that hidden state can take and T is the length of the sequence. The continuous observed variables are represented as $X= (x_1\ldots\ldots x_T)$ $x\epsilon\ \mathbb{R}$. In our case, the hidden state represents the appliance working state (on/off or ex: washing, rinsing), and observed values represent the power demand of that appliance state.

HMM can be defined with three parameters. First, the initial probability of the states that can be in sequence at t=1. It can be represented as vector π such that: $\pi=p(z_1=k)$. Second, the transition probabilities from state i at t-1 to state j at t can be represented with matrix A such that: $A_{i,j}= p(z_t=j|z_{t-1}=i)$. Both initial and transition probabilities follow the categorical distribution. Third, emission probability function Φ for each state which provides the probability of the observed variable is emitted by a certain state at the time t. Generally, the Gaussian distribution function is used for this purpose. The emission probability is represented as such that: $xt|zt, \Phi\sim N(\mu_{zt},\tau_{zt})$. where $\Phi=\{\mu_{zt},\tau_{zt}\}$ and μ_{zt} and τ_{zt} are mean and precision of a state Gaussian distribution. HMM model can be represented as $\theta= (\pi, A, \Phi)$.

The forward-backward algorithm also known as the Baum-Welch algorithm is popular for learning parameters and the Viterbi algorithm is popular for inference purposes. The computational complexity of the exact inference for HMM is $O(k^2T)$ where k is the number of states and T is the number of time steps.

Energy disaggregation using HMM involves training θ. In order to do that we followed the steps proposed by [1]. It involves creating a general model for each appliance using train house data by applying the Baum-Welch algorithm. During the tuning phase, with this general model, it considers the total consumption sequence of the test houses and finds the solo operational period of the appliance of interest. By finding the windows of aggregate data in the total consumption sequence in which most likely only the appliance of interest was operating, it can be used to train the HMM model again with the consumption data from that period. We find that window by running the likelihood estimation algorithm for the small overlapping windows of aggregate data and selecting the top 10 windows period with the most likelihood estimation. The likelihood estimation provides the probability that the sequence is generated by the HMM model. It is possible in that window of data other appliances are providing constant load. We solved that by subtracting the constant baseload in that window of data. the size of the window is determined by the maximum signature length encountered in the training data for the appliance.

2.2.1 Disaggregation using Difference Hidden Markov Model

Once the model is tuned it can be extended to a variant of DHMM by making difference in aggregate power between two consecutive steps as another observation sequence as shown in the Figure 1. In Figure 1, z represents the hidden state, x represents aggregate power demand, y represents the difference in aggregate consumption between steps.

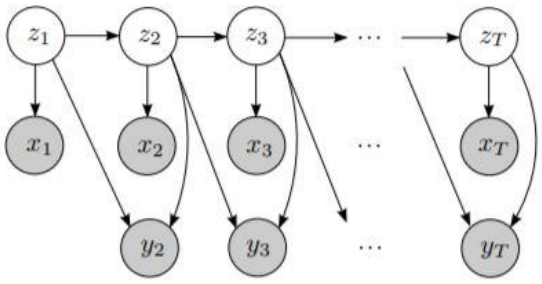

Figure 1: Difference Hidden Markov Model

we represent tuned model parameters as $\theta^{Tuned}= (\pi^{Tuned}, A^{Tuned}, \Phi^{Tuned})$. The probability that step change in aggregate consumption is due to appliance state change is given by the gaussian distributed function. $y_t|z_t, z_{t-1}, \phi \sim N (\mu_{zt} - \mu_{zt-1}, \sigma^2_{zt} + \sigma^2_{zt-1})$. The above functions do not impose the constraint that the appliance can only be on and at a state only if the total consumption is above the mean consumption of the appliance state. In [1], they imposed this constraint using erf function using total consumption reading defined as: $P(wzt \le xt|zt, \phi) =$

$$(\mu zt, \sigma 2zt)\ dw = 1/2\ [\ 1 + erf\ ((xt - \mu zt)/\ \sigma zt \sqrt{2}\ )\ ]\ [1]$$

This function makes the probability of the state tends towards zero if the total consumption reading is much below the mean consumption of the state and vice versa.

Once we have the DHMM model with tuned parameters, any inference mechanism can be used for disaggregating appliance load from total load given

only total load. We used the Viterbi algorithm as it is used by [19]. Viterbi algorithm is extended to process two observations and mean consumption constraints defined above. The Viterbi algorithm evaluates the joint probability of all the sequences in the model x, y, z using the equation given below:

$$P(x, y, z|\theta) = P(z_1|\pi) \cdot \prod_{t=2}^{T} P(z_t|z_{t-1}, A) \cdot \prod_{t=1}^{T} P(w_{zt} \leq x_t|z_t, \phi) \cdot \prod_{t=2}^{T} P(y_t|z_t, z_{t-1}, \phi) \quad [2]$$

The algorithm accepts the window for which joint probability is above a certain specific threshold. Appliance load consumption is assumed to be the mean consumption of the hidden states inferred for the windows accepted. In [1], the disaggregated data is subtracted from total consumption before disaggregating the next appliance to maintain the constraint that total consumption is equal to the sum of individual appliance consumption. In[9], the optimization approach is presented to overcome the disadvantage of the above method. We did not follow any of these two methods to know the algorithm performance. The results are presented and discussed in the results section.

2.3 Factorial Hidden Markov Model(FHMM)

FHMM is an extension of HMM in which there are multiple independent Markov chains. In the NILM context, FHMM is used to model multiple appliances into a single model. Each Markov chain represents one appliance. Our motive for FHMM in this thesis is to explore the possibility of adapting the HMM method above to a set of appliances modelled using FHMM. The reason for this is that collecting submeter data is easier as less sensor is needed or if the dataset from the region only contains sub-metered data for a set of appliances. We could still get valuable information from sub-metered data. If the model is successful, this could potentially act as a virtual submeter. This virtual submeter could improve the individual HMM by providing information or feature.

In the Irise dataset, for electric heating, there is no individual reading but single total electric heating consumption from multiple electric heaters is what available. We also do not know how many electric heaters are there. We developed and tested this FHMM for only Electric heaters. However, the method explored applies to other sets of appliances too. In this work, we are using Additive-DFHMM for unsupervised modelling of a set of ESH and Gibbs sampling for inference. The methodology is explained in the following subsections.

Like HMM general model, FHMM general model was created by selecting the number of chains, the number of hidden states, and prior values. The initial probability of hidden states for our FHMM can be represented as π such that: $\pi = (\pi_m^1 \ldots \Pi_m^N)$ where m represents the number of hidden states and N represents the number of chains. Each element in the π represents the initial probability of states in an individual chain and follows the categorical distribution. The transition probability of hidden states for our FHMM can be represented as A such that: $A_{i,j} = (A_{i,j}^1 \ldots\ldots A_{i,j}^N)$. Here i, j represents the state at t-1 and t. N represents the number of chains. Each element in $A_{i,j}$ represents the transition probability matrix for individual chains and follows the categorical distribution. The emission probability function for our FHMM observed variable can be represented as such that: $x_t|z_t^{(1:N)} \sim N\left(\sum_{i=1}^{N} \mu_{zt(i)}^{(i)}, \sum_{i=1}^{N} \tau_{zt(i)}^{(i)}\right)$. We used Gaussian distribution to model each state's emissions and the summation operator in the above function indicates that the observed variable is the sum of the state emissions from each chain at a particular time.

To create a general model, we followed the same steps we described in HMM general model section. For each chain, we created a general chain by drawing samples from corresponding chain parameters estimated for each house. Finally, we combine the general chains to create a general model. Tuning the general model to an individual house also follows the same steps we described in the tuning general model section. Once the model is tuned, we can use ADFHMM for disaggregation which we explained in the below subsection.

2.3.1 Additive Difference Factorial Hidden Markov Model (ADFHMM)

FHMM can be extended into ADFHMM by adding another observation variable that represents the difference in aggregate consumption between two consecutive steps. The graphical model of ADFHMM is shown in Figure 2.

The probability that step change in aggregate consumption due to state changes in the chains is given by the Gaussian function $y_t|z_t^{(1:N)}, z_{t-1}^{(1:N)} \sim N\left(\left(\sum_{i=1}^{N} \mu_{zt(i)}^{(i)} - \sum_{i=1}^{N} \mu_{zt-1(i)}^{(i)}\right), \left(\sum_{i=1}^{N} \tau_{zt(i)}^{(i)} + \sum_{i=1}^{N} \tau_{zt-1(i)}^{(i)}\right)\right)$

Since the exact inference for ADFHMM is computationally intensive, We used Gibbs sampling for inference to disaggregate ESH consumption from total consumption reading. The joint probability of all sequences in the model z1, z2, x,y using the equation below. $p(x, y, z^{(1:N)}|\theta) =$

$$\prod_{i=1}^{N} p(z_1^{(i)}|\pi^{(i)}) \prod_{i=1}^{N} \prod_{t=2}^{T} p(z_t^{(i)}|z_{t-1}^{(i)}, A^{(i)}) \prod_{t=1}^{T} p(w_{zt} \leq x_t|z_t^{(1:N)}, \phi) \prod_{t=2}^{T} p(y_t|z_t^{(1:N)}, z_{t-1}^{(1:N)}, \phi) \quad [3]$$

The term $P(wzt \leq xt|zt(1:N), \phi)$ is the erf function similar to the one used in the HMM model to impose the constraint that certain state combinations are not

possible unless the aggregate consumption is above the sum of mean values of the states of each chain.

Based on the appliance-specific threshold, the algorithm accepts a window of data with a probability higher than the specified threshold.

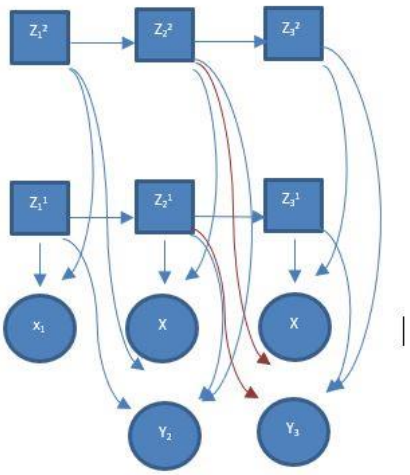

Figure 2:ADFHMM

The advantages of this method are that it requires less training time and data, it could act as a virtual submeter, and can provide additional information for the HMM model. The disadvantages are it is computationally very expensive, it cannot specifically give individual appliance consumption. It is also prone to noise during tuning and disaggregation.

2.4 Variational AutoEncoder (VAE)

An autoencoder is a Deep Neural Network (DNN) model that is trained to produce an output closer to the input with the minimum loss[10]. the main purpose of the autoencoder is to compress the data efficiently by finding the key features. AutoEncoder (AE) variant Variational autoencoder was initially proposed in [11] in which latent space are regularised. in VAE encoder outputs are not deterministic as it is in AE. Instead, in VAE the encoder outputs are probabilistic distributions. We used Gaussian distribution to represent input data in latent space. Multiple variants of VAE, training methods are proposed depending upon the application. We use simple VAE for NILM.

The Kullback-Leibler divergence between two probability distributions p and q can be given by $D_{KL}(p||q) = \sum_{i=1}^{N} p(xi) \cdot (\log p(xi) - \log q(xi))$ .The loss function used for VAE training can be represented as Loss=|| x-$\hat{x}$||^2 + KL [N(μ_x , σ_x), N(0,I)] .

The principle used in this thesis for load classification is to extract features using a VAE encoder and classify them using Neural Network (NN) based classifier. At first, VAE is trained with the appliance data from train houses. Once VAE is trained, by taking only the encoder part and fixing the weight, it can be used as a feature extractor. Here the features are mean and variation of the distributions in latent space.

NN based classifier is trained with extracted features and labels to classify loads in a supervised manner. Any Classifier algorithm can be trained on top of VAE. Once the training is finished, during testing, we used an encoder to extract features and a classifier to classify the sequence for the test houses.

For each appliance, individual VAE and NN classifiers are trained. The training data is prepared with a certain overlapping window of appliance data. The window length was decided based on the data encountered in the training set. The overlapping window was decided by multiple testing with different window lengths for each appliance and select the one that provides minimum loss function value. The data is normalized to bring the training data to a common scale.

For the VAE of each appliance, network architecture, number of layers, number of neurons, type of neurons, number of distributions for latent space were decided by testing multiple configurations along with multiple configurations for classifier and selecting the one that provides minimum loss function value and higher classifier accuracy. We considered Long-Short Term Memory (LSTM), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN) for the type

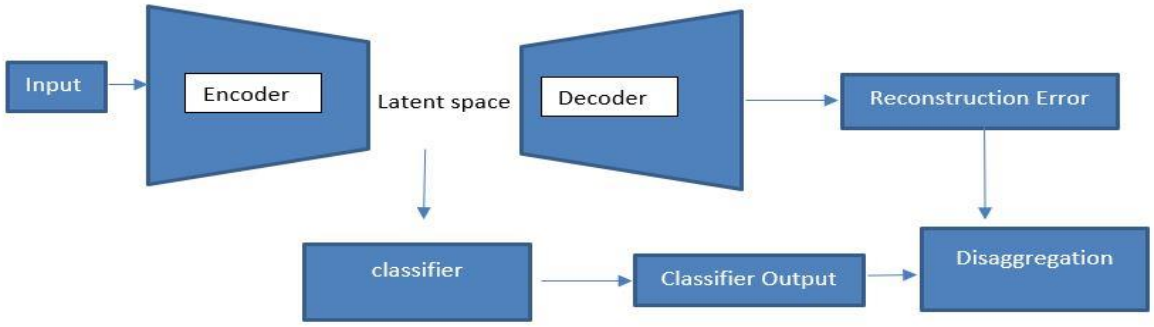

Figure 3:VAE based NILM

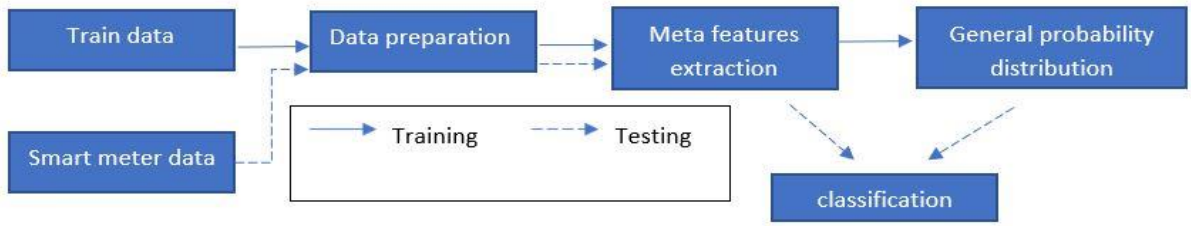

Figure 4:Statistical Rules based classification for NILM

of layers. LSTM and RNN are popular for their ability to model sequential data and CNN is popular for pattern recognition ability. We used CNN for NN based classifier due to its recent popularity in pattern recognition, object detection, classification, and other applications. The final layer of the CNN classifier is the discriminative layer. This layer provides the probability of the class for the given input.

The steps involved in the VAE based NILM is shown in Figure 3. The algorithm accepts the window if it satisfies the following conditions: the probability output of the classifier is above a certain specific value and the reconstruction error of the VAE is less than a certain specific value. These specific values are decided by testing the model with training data and select the one that provides maximum accuracy. It should be noted that for the selected value, the model might have low accuracy for test houses than train houses. This is because the model is trained with train house data. for test houses, the entire process is unsupervised. The reason that reconstruction error is included for evaluating the window is that VAE performs poorly in reconstructing the input data if it has not seen similar data in training. We also included TOU into VAE. We used weightage for final probability calculation which can be given as W1xVAE output+W2xTOU where the sum of W1 and W2 is 1.

The advantages of this method are feature extraction is automatic, which can be used to generate synthetic data and can perform in real-time. The disadvantages are it requires a long training time, a large amount of data for training, training data should be general enough for the appliance, cannot be tuned to a specific house.

2.5 Temporal classifier based on statistical rules

This method involves attribute construction by knowledge extraction of the substructures in the training instance. By providing additional attributes, propositional learners can perform better than without these attributes. These substructures are called meta-features. Indeed, the choice meta-features depend upon the problem and application. In our case, meta-features are based on time and consumption values.

The steps involved in this method for load classification is as follows: Subsequences are generated using a temporal sliding window from the training data. the window length and sliding width are decided based on the data encountered in the training dataset. Data is normalized. Meta-features are generated for the subsequences. Generic meta-features are developed for each appliance. This step involves generalizing the meta-features obtained for the specific appliance from different train houses so that it can represent all appliances of a specific type. This step is similar to the HMM generic appliance model creation. We introduced this step to make the supervised method unsupervised for test houses. This step is explained in detail in generic meta-features creation. Using generic meta-features, instances of test houses are classified. The steps involved in the Statistical rules-based NILM is shown in Figure 4.

2.5.1 Meta-features

Time of use

This feature will represent which time-period appliance has been used. Knowledge extractor counts the number of readings (appliance on) in each time period in the event window. The time of use can be represented as TOU(h)=N; Nϵ R, hϵ(1,24) Where N represents the number of above zero value readings. h represents the hour.

Day of the week (DOW)

This feature represents on which day of the week the appliance has been used and can be represented as below: DOWn= D; Dϵ (0,6)

Max and min values positions

This feature represents the positions of the max and min values in the event window. If there is two or more max or min values. This feature will provide their positions also. It can be represented as below.

$$Pos_{max} \longrightarrow E(pos_{max})=\max(E_i);\ i\epsilon(1,L) \quad [4]$$

$$Pos_{min} \longrightarrow E(pos_{min})=\min(E_i);\ i\epsilon(1,L) \quad [5]$$

Where E_i represents energy value at the i^{th} position in the window, i represents the position in the window, L represents the length of the window.

Energy difference

This feature represents the energy difference of each step with respect to the certain reference in the instance. $E_{diff}(p) = E(p_{ref}) - E(p);\ p \in [0, L]$. Where p, L,

and E represents the position in the window, window length, the energy value at a position.

Gradient, Gradient ratio, Laplacian

Gradient ∇ represents a change in energy consumption between the current timestep and the previous time step. The gradient is calculated for all time steps in the window except first. Laplacian Δ represents a change in the gradient between the current timestep and the previous time step. Gradient ratio ∇_r represents the ratio between the gradient and the energy value at any given time step. These features provide trends in energy consumption in the event window.

$$\nabla(p) = E(p) - E(p-1) \quad [6]$$

$$\Delta(p) = \nabla(p) - \nabla(p-1) \quad [7]$$

$$\nabla_r(p) = \nabla(p)/E(p) \quad ; p \in [0, L] \quad [8]$$

the features extracted from training data are used to construct generic value and probability distribution for these meta-features that can represent any appliance of a specific type.

TOU, DOW, max, and min value positions are discrete variables. The data from each event window is added to create an overall value for these three parameters. It is possible to create categorical distribution to each house with the knowledge extracted. Once it has been extracted, it is possible to create a generic categorical distribution for TOU by sampling from each distribution and fitting the categorical distribution to the samples. Individual distribution is needed here to avoid the domination of data from the house in which the appliance has been used much higher than other houses.

Energy difference, gradient, gradient ratio, Laplacian are continuous variables. For each house, a general model needs to be created that represents all event window. This is achieved by constructing Gaussian distribution for all the variables of these features using the knowledge extracted. After creating distributions for each house, general distribution can be created similarly by drawing an equal number of samples from each distribution and fitting the model to samples.

Once general distributions are created, it can be used to classify sequences from unforeseen houses. This is achieved by first extracting the same meta-features in the sub-sequence and evaluating the probability using a general probability distribution. The subsequence with the final probability above a certain specific appliance threshold is accepted by algorithm. The final probability is calculated using the equation given below:

$$P_{final}=P_{TOU\_DOW}\times P_{MaxPos}\times P_{MinPos}\times P_{ED}\times P_G\times P_{GR}\times P_L \quad [9]$$

$$P_{TOU}(H)=(P(H-1)+P(H)+P(H+1))/3 \quad [10]$$

$$P_{TOU\_DOW}=W1\times P_{TOU}+W2\times P_{DOW} \quad [11]$$

P_{TOU_DOW} represents the time of use probability and day of week probability. P_{TOU} represents TOU probability for the hour index H. Where W1, W2 are weightage and its sum is equal to 1. In this way, the algorithm can include domain knowledge and at the same time robust to the occupant's habits.

Since some of these meta-features have more than one variable, the algorithm calculates the probability for all the variables of the meta-feature and takes the average out of it. The advantages are it is simple, interpretable, requires less training time computationally less intensive than HMM and FHMM. The disadvantages are it is prone to noise and less efficient for medium-low consuming multi-state devices, the features need to be distinct enough.

2.6 Novel postprocessing

Our work is based on the bottom-up approach proposed [12] that is based on the consumption behaviour of the residents. The idea proposed by the authors is by extracting similar days to the day of interest and evaluating appliance consumptions on those historical days it is possible to find the probability of the appliance being operated on the day of the interest.

Our hypothesis to this method is by extracting similar time windows of the test data for an instance and by evaluating its labels provided by any disaggregation algorithm it is possible to detect the wrongly labelled time window to a certain extent. We propose a novel post-processing method especially for low-frequency unsupervised algorithms with no special data requirements to improve the NILM.

A similar STW concept is proposed in [13] for NILM. they used a supervised method to obtain the energy consumption of the device. Our algorithm is different because we do not use any ground truth data. Our algorithm relies on the labels labelled by the NILM classification algorithm. To explain in simple words, if my eight out of the ten neighbours are blue then I am most likely blue than red.

The graphical representation of the steps involved in this method is shown in Figure 5. Our algorithm starts with the initial event window and looks for similar time windows in the consumption sequence. The similarity is measured using the Euclidean distance measure between the features extracted from the test window and all other windows in the sequence. Smaller distance means features are similar. Once it obtains K-STW's, it counts the labels assigned to these windows by the NILM classifier. The algorithm ignores the test

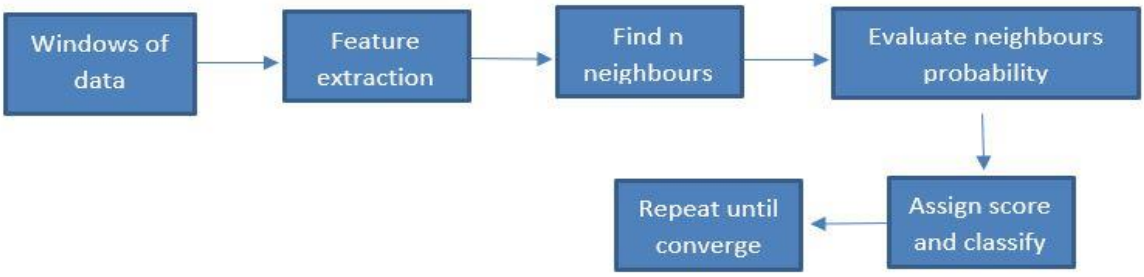

Figure 5:STW post-processing

window's label as it is the test point. The label of the test window is assigned based on the following conditions. First, the label count should be above a certain threshold. Second, the probability assigned to the test window by the NILM classifier should be at least 75% of the threshold probability of the appliance used by the NILM classifier. Similarly, the algorithm iterates over all windows and post-process the labels. The algorithm is repeated until it converges. We applied this postprocessing to all three methods and compared the results.

2.6.1 Features extracted

Time of use

It provides the first time-stamp hour value in the event window. It can be represented as TOUn=H; Hϵ (1,24)

Day of the week

It provides a day of the week value of the event window. Day of the week can be represented as below: DOWn= D; Dϵ (0,6)

The month of the window

This provides the month value of the event window. It can be represented as Mn=m; mϵ (1,12)

Mean value

It provides the mean of the values in the event window. It can be represented as below: $\mu_n \epsilon R$

Number of elements greater than a certain value

It provides the number of elements greater than a certain value in the event window. It can be represented as $N_n \epsilon R$

Dynamic time warping (DTW) distance

DTW is a popular distance-based similarity measure between two-time series. Similarity measure based on the features defined above between test point X_t and any other point X_n can be defined as below:

$$SM(X_t,X_n) = \sqrt[2]{(TOUt - TOUn)^2 + (DOWt - DOWn)^2 + (Mt - Mn)^2 + (\mu t - \mu n)^2 + (N_t - N_n)^2 + (DTWt,n)^2} \quad [12]$$

All the variables were normalized to equal out the influence of each variable. The advantages are that it improves the accuracy of disaggregation in addition to the classifier's maximum performance. The disadvantages are it relies on classifier accuracy and computationally expensive.

3.Results

We analysed 50k data points for each house among which 13k-30k represents the summer consumption. The results presented here are the average results of test houses due to space constraints. It is not fair to compare average (Avg) results because sometimes the model could perform very high for one house but very poor for other houses yet can have a decent average.

Water heater

Table 1:Performance results of different models for water heater

Avg	HMM general model	Hmm tuned model	VAE	VAE +TOU	Stats rules
Pre	0.49	0.54	0.72	0.72	0.52
Rec	0.75	0.57	0.75	0.75	0.53
F1	0.57	0.49	0.72	0.72	0.49

Table 2:Performance results of different models for water heater during summer

Avg (summer)	HMM tuned model	VAE+TOU	Stats rules
Pre	0.84	0.88	0.71
Rec	0.58	0.73	0.44
F1	0.68	0.79	0.45

The disaggregated results of the water heater are shown in tables 1 and 2. for HMM, the general model performs better than the tuned model. However, the individual analysis on each home was not conclusive. VAE performs better than other models and TOU does not improve the VAE model. the individual analysis showed that TOU improves results. However, tuned TOU might be consistent that General TOU. The statistical model performs similarly to the tuned model. Except for the statistical model, summer disaggregation is better than overall disaggregation. Summer values for the statistical method indicate that the model has different issues than the other two models.

Space heater

The disaggregated results for the space heater are presented in Table 3. FHMM performs better than VAE.

Table 3: performance results of different models for Space heater

Avg	FHMM General model	FHMM Tuned model	VAE	VAE +TOU	Stats rules
Pre	0.73	0.76	0.83	0.8	0.68
Rec	0.85	0.76	0.48	0.49	0.8
F1	0.78	0.76	0.61	0.61	0.74

However, VAE has the highest Precision. The statistical model performs similarly to the FHMM model.

Washing machine

The disaggregated results of the washing machine are shown in Tables 4 and 5. Overall all three models were poor. VAE and statistics method performs better than the HMM method. Summer disaggregation is better for all three models than overall disaggregation. The statistics method has better recall than any other models. TOU does not improve VAE's performance.

Table 4: performance results of different models for washing machine

Avg	HMM general model	Hmm tuned model	VAE	VAE +TOU	Stats rules
Pre	0.06	0.1	0.17	0.17	0.13
Rec	0.11	0.2	0.28	0.28	0.40
F1	0.07	0.1	0.19	0.20	0.18

Table 5: performance results of different models for washing machine during summer

Avg (summer)	HMM tuned model	VAE+TOU	Stats rules
Pre	0.19	0.25	0.18
Rec	0.24	0.24	0.35
F1	0.20	0.23	0.23

Dishwasher

The disaggregated results of the dishwasher are presented in Tables 6 and 7.

Table 6: performance results of different models for Dishwasher

Avg	HMM general model	Hmm tuned model	VAE	VAE +TOU	Stats rules
Pre	0.085	0.09	0.20	0.20	0.07
Rec	0.18	0.17	0.26	0.26	0.18
F1	0.11	0.1	0.22	0.22	0.11

Table 7: performance results of different models for Dishwasher during summer

Avg (summer)	HMM tuned model	VAE+TOU	Stats rules
Pre	0.13	0.26	0.13
Rec	0.21	0.21	0.135
F1	0.15	0.23	0.12

All three models were poor. The model's performance was similar to the washing machine with VAE being the better model. Individual houses' analysis showed that the VAE of the dishwasher is actually better than VAE of washing except for house 14.

Clothes dryer

The disaggregation results for the clothes dryer is shown in Table 8. Though the recalls were better for clothes dryer than washing machine and dishwasher. The precision value is very poor. The models could not distinguish other appliances.

Table 8: performance results of different models for clothes dryer

Avg	HMM general model	Hmm tuned model	VAE	VAE +TOU	Stats rules
Pre	0.08	0.07	0.11	0.11	0.076
Rec	0.24	0.2	0.34	0.34	0.43
F1	0.11	0.1	0.17	0.17	0.12

Postprocessing(pp)

Table 9: Post-processing results for water heater VAE

VAE+ TOU 13k-30k	H14 before PP	H14 after PP	H28 before pp	H28 after PP	H36 before pp	H36 after PP
Pre	0.94	0.87	0.87	0.97	0.84	0.86
Rec	0.61	0.81	0.76	0.98	0.83	0.95
F1	0.74	0.83	0.81	0.97	0.83	0.9

we applied our postprocessing method to the VAE results of the water heater and are presented in Table 9. In Table 9 one can see the overall improvement in the F1 score for all three houses. Particularly for house 28, the improvements are remarkable. F1 score increases from 0.81 to 0.978. One can see there is a decrease in precision score for house 14. This is due to sometimes if the NILM algorithm accepts similar false windows our postprocessing method can not know that and it will accept more similar false windows.

Table 10:postprocessing results for washing machine VAE

VAE+ TOU 13k-30k	H14 before PP	H14 after PP	H28 before pp	H28 after PP	H36 before pp	H36 after PP
Pre	0.19	0.2	0.09	0.12	0.46	0.4
Rec	0.35	0.67	0.09	0.27	0.30	0.49
F1	0.25	0.31	0.09	0.17	0.36	0.43

To show our method applies to all appliances, we applied our method to washing machine VAE results and are presented in Table 10. One can see improvements in the F1 score for all houses even when their F1 score is below 0.5. In particular, house 28 results show the strength of our proposed method. Even when precision and recall were below 0.1 our method was able to improve the results.

To show that our method is common to all NILM algorithms, we applied the postprocessing to House 14 water heater results of all three methods explored in this work are presented in Table 11. There is an improvement in the F1 score for all three methods.

Overall, it can be said that our postprocessing method can improve the results. It can also be interpreted from all three tables that postprocessing results depend upon the appliance of interest and other appliances' characteristics in the house and also NILM algorithms capability. One can also see there is not much improvement in precision and sometimes precision score decreases after postprocessing. The reason is due to false positives labelled by the NILM, and we will focus on this problem in future works.

Table 11:Postprocessing results for house 14 water heater-different methods

House 14 13k-30k	VAE before PP	VAE after PP	HMM before pp	HMM after pp	Stats rules before pp	Stats rules after pp
Pre	0.94	0.87	0.89	0.82	0.87	0.8
Rec	0.61	0.81	0.48	0.71	0.26	0.71
F1	0.74	0.83	0.62	0.76	0.40	0.75

4.Conclusion

We explored three different models for NILM and tested for different appliances. In summary, the results showed that models can be deployed in the real-world for demand-side management only for the water heater and electric space heating. The methods can not be completely non-intrusive and high performing. All three methods performed poorly for multi-state devices like washing machines, dishwashers, clothes dryers. One of the methods we proposed VAE for NILM showed similar performance for the water heater and electric heating and higher performance for multi-state devices than the state of the art HMM method for NILM. Another method we proposed also shows similar performance for the water heater and electric heating to the other two methods. When it comes to multi-state devices the model did not perform better than VAE. We validated our hypothesis for postprocessing and proved that results can be improved. Our post-processing method's ability to improve Recall more provides flexibility to the NILM algorithm to tighten the threshold thereby it can reduce the false positives.

Since we treated each appliance model-independent and applied total energy as input the same window has been accepted by multiple appliance models. In the future, we plan to explore methods to enforce the constraint that a single-window can belong only to one model. We tested FHMM only for space heaters. We plan to test it for a different set of appliances and validate its ability to act as a virtual sub-meter and explore the possibility to use it as a feature generator for further disaggregation. Since VAE is known for its new content generation, we will explore the ways to use this for NILM benefits. Possibly to get more training data or with some occupant's survey, it can create the possible signal dictionary for the appliance in the house. We will also explore the possibility of reducing the training time and include domain knowledge into the VAE model. Our statistical rules-based method is very naïve and results showed that features were not distinctive enough. We plan to explore more features to improve the results. We also plan to incorporate domain knowledge and study feature importance. Postprocessing can be further improved by including more features and domain knowledge like appliance correlations, appliance usage patterns. For example, it might not be possible to have a washing machine being used more than 15 times in a week. This kind of knowledge can be used to further improve the results.

References

[1] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Proceedings of the National Conference on Artificial Intelligence*, 2012.

[2] W. He and Y. Chai, "An Empirical Study on Energy Disaggregation via Deep Learning," in *2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE 2016)*, 2016.

[3] M. D'Incecco, S. Squartini, and M. Zhong, "Transfer Learning for Non-Intrusive Load Monitoring," *IEEE Trans. Smart Grid*, 2020.

[4] W. Xu, H. Sun, C. Deng, and Y. Tan, "Variational autoencoder for semi-supervised text classification," in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 2017.

[5] E. Karamatli, A. T. Cemgil, and S. Kirbiz, "Audio Source Separation Using Variational Autoencoders and Weak Class Supervision," *IEEE Signal Process. Lett.*, 2019.

[6] K. Basu, V. Debusschere, and S. Bacha, "Residential appliance identification and future usage prediction from smart meter," in *IECON Proceedings (Industrial Electronics Conference)*, 2013.

[7] L. Pereira and N. Nunes, "Performance evaluation in non-intrusive load monitoring: Datasets, metrics, and tools—A review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. 2018.

[8] "irise." [Online]. Available: https://remodece.isr.uc.pt/.

[9] M. Figueiredo and A. De Almeida, "On the Optimization of Appliance Loads Inferred by Probabilistic Models," *NILM Work. 2014*, no. 1, 2014.

[10] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, 1989.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[12] B. Gao, X. Liu, and Z. Zhu, "A bottom-up model for household load profile based on the consumption behavior of residents," *Energies*, 2018.

[13] X. Shi, H. Ming, S. Shakkottai, L. Xie, and J. Yao, "Nonintrusive load monitoring in residential households with low-resolution data," *Appl. Energy*, 2019.