TreeCNN and NILMTK Unite: Illuminating Energy Efficiency in Real-World Scenarios

Sabiha Afroz
Computer Science Department
Virginia Tech, Blacksburg, USA
sabihaafroz@vt.edu

Buvana Ramanan, Manzoor Khan
Autonomous Systems Research Department
Nokia Bell Labs, Murray Hill, USA
{buvana.ramanan, manzoor.a.khan}@nokia-bell-labs.com

Ali R. Butt
Computer Science Department
Virginia Tech,
Blacksburg, USA
butta@cs.vt.edu

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10825584

***Abstract*— Efficiently managing electricity supply and demand, especially during peak times to minimize waste, remains a key challenge for the electric grid. An effective solution involves incentivizing users to shift their shiftable loads, such as dishwashers and washing machines, to off-peak periods. Non-Intrusive Load Monitoring (NILM) provides a cost-effective and pragmatic approach for detailed appliance energy consumption insights. Among Deep Learning models, TreeCNN has shown superior performance compared to RNN and traditional CNN models in energy disaggregation. However, its evaluation has been limited to the Dataport dataset. To fully assess TreeCNN's capabilities, comprehensive testing with diverse datasets like REDD, UK-DALE, DRED and others is essential. Additionally, integrating TreeCNN into NILMTK, a dataset standardization tool, enables thorough comparisons with 16 formatted datasets and other disaggregation algorithms. In this work, we integrated TreeCNN into NILMTK toolkit and benchmarked, providing valuable insights into its effectiveness and real-world usability.**

***Index Terms*—Non-Intrusive Load Monitoring, load disaggregation, smart energy management, demand response, Artificial Intelligence, Machine Learning, Deep Learning, TreeCNN, NILMTK, CNN, smart meter, Sequence-to-Point, REDD, UK-DALE**

I. Introduction

The world is currently facing energy shortages [1] and carbon emission problems [2]. Statistics show that, on average, 20-30% [3] of a building's energy is wasted. However, research studies indicate that optimal use of electric appliances can reduce energy consumption by up to 20% [4]. Consumers cannot effectively reduce energy usage just by looking at monthly electricity bills. To understand the electricity consumption of each household appliance, they need detailed consumption data for each device. Therefore, detailed analysis reports can motivate users to change their behavior regarding unnecessary electricity consumption. Additionally, old or faulty appliances consume more electricity, increasing bills. In this scenario, aggregated monthly bills do not help clients identify inefficient appliances which contribute most to electricity costs. Hence, disaggregated energy consumption data would help consumers identify malfunctioned appliances and replace them with new ones.

Due to the increasing demand [5] for electricity, grid operators face challenges in load balancing. To address this, many countries are installing smart meters [6] to obtain detailed electricity usage patterns. With this detailed data analysis, power suppliers can generate accurate predictions. Consumers can use these predictions to adjust their behavior and use electricity more efficiently. Moreover, power companies can also incentivize clients for optimal usage. For example, consumers could shift their electricity usage to off-peak hours, earning incentives from power companies and reducing their electricity bills. Electricity bills are significantly higher during peak hours, so shifting usage can result in substantial electricity savings. Therefore, it is very crucial to break down aggregate energy to appliance specific usage.

There are two approaches to load monitoring [7], [8]: 1) Intrusive Load Monitoring (ILM) and 2) Non-Intrusive Load Monitoring (NILM). ILM measures each appliance's energy consumption by attaching sensors to each device. Although this method provides accurate readings, it is costly and requires high maintenance. Conversely, the NILM method [9] calculates disaggregated loads for each appliance using the total electricity consumption data from a smart meter. Hart [9] first introduced the NILM approach. While NILM is less accurate than ILM, it is more cost-effective and avoids ILM's disadvantages. Consequently, NILM is gaining popularity within the research community.

Combinatorial Optimization (CO) [10] is an early algorithm used for NILM. It aims to find the optimal set of operational states that best reconstruct the total power consumption of a house. However, CO is susceptible to transients, an increasing number of devices, and devices with similar characteristics. State-based mathematical models, such as the Factorial Hidden Markov Model (FHMM), have shown notable performance for NILM [11]. FHMM analyzes aggregate power signals to estimate the hidden operational states of each appliance, considering their state continuity over time [12]. While FHMM-based methods effectively disaggregate periodic loads, they perform less well with appliances that have short or infrequent operating cycles [7]. Additionally, state-based models incur high computational costs if a household has many appliances

with many power states [13].

Deep Neural Networks (DNNs) excel in natural language processing, computer vision, and speech recognition. Recognizing the potential of DNN models for NILM, researchers adapted convolutional neural networks (CNN) [14] and long short-term memory (LSTM) models [15] for load disaggregation, achieving better results compared to mathematical models [8]. C. Zhang [16] proposed a Sequence-to-Point (Seq2Point) neural network based on CNN, which outperformed existing state-of-the-art models. The Seq2Point method is a real-time energy disaggregation technique that uses a sliding window of aggregate energy to predict the midpoint value of appliance consumption within that window [17].

Smart meter specifications worldwide indicate that most smart meters sample data at an hourly rate [18]. Therefore, source-separation algorithms must process low sampling rate time series data. Y. Jia [19] proposed a tree-structured neural network model based on CNN, which outperformed Recurrent Neural Network (RNN) [20] and traditional CNN models in energy disaggregation on low frequency data. However, the authors used only the Dataport [21] dataset for benchmarking. This raises several questions: How does the TreeCNN model compare with other state-of-the-art models like Seq2Point [16]? Can it generalize across different publicly available datasets such as REDD [22] and UK-DALE [23]?

NILMTK [24] is a popular open-source toolkit that helps researchers compare new algorithms with existing energy disaggregation algorithms, reproduce experimental results, and compare across different datasets in a standard form. This toolkit also provides various measurement metrics. We aimed to incorporate the TreeCNN disaggregation model into this toolkit to benchmark it against existing models and facilitate future comparisons with new algorithms. Benchmarking TreeCNN across different datasets and models will help us understand its generalization capabilities and performance with different types of appliances.

Our main contributions are:

- Integrating the TreeCNN model into the open-source NILMTK toolkit. For this integration, we preprocessed the time series data into higher-dimensional data suitable for the TreeCNN model.
- Comparing the performance of TreeCNN with three energy disaggregation models: 1) CO, 2) FHMM, and 3) Seq2Point.
- Evaluating the performance of the TreeCNN model with five public datasets: 1) REDD, 2) UK-DALE, 3) DRED 4) Smart*, and 5) IDEAL.

This paper is organized as follows: Section II provides a detailed overview of various existing NILM models. Section III outlines the steps for integrating the TreeCNN model into the NILMTK toolkit. Section IV discusses the dataset selection, appliance selection, evaluation metrics, experimental setup, and results. Section V points out some limitations of TreeCNN model, scope for improvement and comparative analysis with Seq2Point. Finally, Section VI summarizes the findings and suggests directions for future research.

II. Background & Motivation

A. What is Energy Disaggregation?

With the disaggregation approach, we can collect the entire building's energy usage and then determine the energy consumption for each appliance, such as HVAC units, dishwashers, and refrigerators. Knowing each appliance's energy consumption has many advantages. Residents can estimate each appliance's contribution to their electric bill and identify any malfunctioning appliances. This feedback can also motivate behavioral changes. For example, since electricity prices are higher during peak hours, users could run heavy appliances during off-peak hours. Electricity distributors, after observing the forecast, can offer incentives to users to shift their workload to off-peak hours, benefiting both parties. As energy is not an unlimited resource, this approach helps reduce energy waste. The recent boom in smart meters is creating opportunities for better energy disaggregation.

B. Combinatorial Optimization (CO)

G. Hart [10] first introduced the CO algorithm for non-intrusive load monitoring. Suppose there are n appliances. Let $x(t)$ be an n-component Boolean vector representing the state (ON or OFF) of the n appliances at time t. The aggregated power at time t is the total sum of the power of individual appliances that are ON. This problem can be formulated as a CO problem (1) [10]:

$$\hat{x}(t) = \arg\min_{x} |P(t) - \sum_{i=1}^{n} x_i P_i| \tag{1}$$

In this context, P_i denotes the p-vector representing the power consumed by the ith appliance during its operation, while $P(t)$ refers to the p-vector of power at time t. The objective is to minimize the difference between the observed power and the predicted power. This is an NP-complete "weighted set" problem that can only be solved by exhaustive search when the size of n is small. This approach struggles to identify simultaneous state changes of multiple appliances [7].

C. Factorial Hidden Markov Model (FHMM)

The load disaggregation problem can be modeled as a Hidden Markov Model (HMM) [25], [26]. An HMM consists of two types of variables: observed variables and hidden variables. The sequence of hidden variables forms a Markov process, where hidden variables represent each appliance's state (ON, OFF), and observed variables represent power usage in NILM. A FHMM is more suitable for modeling time series generated by several independent processes. FHMM [27] is a generalization of HMM, featuring multiple independent hidden state sequences, with each observation depending on multiple hidden variables. To design the model, we need four components: 1) A finite set of hidden states $x = x_1, x_2, x_3, ..., x_N$, 2) A transition matrix t, representing the probability of changing states, 3) An emission matrix e, representing the probability of emitting an observation, 4) An initial state probability distribution $\pi = \{\pi_i\}$.

D. Sequence-to-Point(Seq2Point)

Kelly and Knottenbelt [20] proposed the Sequence-to-Sequence (Seq2Seq) model in 2015, which improved performance compared to existing machine learning models. In 2017, C. Zhang [16] introduced the Seq2Point model based on CNN, which outperformed Seq2Seq. Seq2Point model takes a mains window $Y_{t:t+W-1}$ and outputs the midpoint element x_m of the corresponding window for the target appliance, where $m = t+\lfloor W/2 \rfloor$. Seq2Point generates a single prediction for each step. For each time step prediction, the model must iterate through each sample in the window, leading to multiple predictions for the same data input. This approach requires significant calculations and computational power, making the process time-consuming [17].

E. TreeCNN

NILM algorithms work with high frequency (MHz range), mid frequency (KHz range), and low frequency (less than 1 Hz) data. While high frequency data captures the entire signal, its collection is costly [17], [28]. However, the widespread deployment of smart meters has facilitated access to low frequency data. Y. Jia [19] proposed the TreeCNN NILM model for low frequency data. This DNN model uses CNN to capture regular temporal energy consumption patterns in households. The tree structure of the algorithm isolates the pattern learning of each appliance, avoiding magnitude variance problems and separating known from unknown appliances. This model outperformed low frequency models like RNN and basic CNN models. RNNs suffer from vanishing gradient problem and lack parallel computation capabilities [16], [29].

Different appliances exhibit distinct temporal patterns. For example, microwaves follow an hourly usage pattern during meal times, while dryers display a daily pattern due to periodic use. Modeling hourly time-series data as a one-dimensional sequence is insufficient to fully describe appliances. Instead, this data should be viewed as a high-dimensional compound of various temporal patterns. The TreeCNN model uses CNN to create spatial filters from hourly energy data, seen as a 2-D matrix. These filters distinguish appliances with unique temporal patterns. For instance, filters for microwaves emphasize the hourly dimension, for dryers the daily dimension, and for HVAC a combination of both. This allows aggregate readings to be projected into their corresponding appliance usage.

The tree structure of the TreeCNN model iteratively decomposes each appliance's energy consumption from the aggregate energy. Each node of the TreeCNN model is a CNN model. During the convolution phase, the CNN model takes the aggregate input and reduces it to a denser representation, extracting temporal patterns from the sparse and granular data, resulting in a lower-dimensional, denser matrix. In the deconvolution phase, the decoder reverses the encoder's actions to reconstruct the input. So, the root node of the tree takes aggregate energy data as input and regenerates its associated appliance's energy consumption as output. The difference between the input and output is then passed down to the child node of the tree as input to reconstruct the next associated appliance's energy consumption. Large energy-consuming appliances overshadow low energy-consuming appliances, so placing these low energy-consuming appliances at the end of the tree alleviates this issue. Additionally, unknown energy consumption introduces errors in energy disaggregation. Therefore, the tree structure of this model views unknown energy consumption as a special appliance to give a more accurate estimation of the observed appliances. Since finding the optimal tree structure is an NP-complete problem, the authors of the TreeCNN paper [19] employed a greedy approach to determine an optimal tree structure.

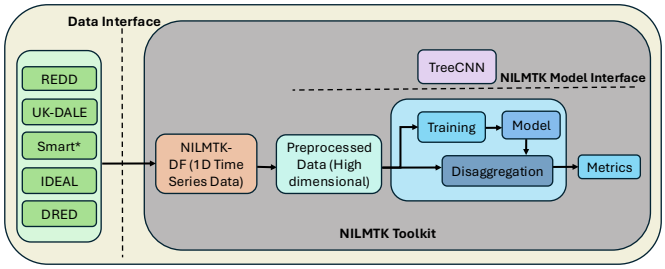


Fig. 1: TreeCNN model incorporation pipeline in NILMTK toolkit.

F. Non-Intrusive Load Monitoring Toolkit (NILMTK)

In 2014, N. Batra proposed an open-source toolkit called the Non-Intrusive Load Monitoring Toolkit (NILMTK) for the NILM research community [24]. Implemented in Python, NILMTK provides a complete pipeline from datasets to accuracy metrics. Publicly available datasets come in various formats, making it difficult for researchers to compare their algorithms across different datasets. NILMTK addresses this issue by using a standard NILMTK-DF data format and includes sixteen data converters to standardize public data.

NILMTK also implements benchmarking algorithms such as CO, FHMM, Seq2Point, and Seq2Seq, allowing researchers to compare their algorithms with state-of-the-art models. This has improved the reproducibility of experimental results. Additionally, NILMTK offers standard accuracy measurement metrics, including Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and F1 score.

The toolkit follows a modular structure and provides APIs for detailed analysis of each dataset, enhancing understanding. It is also well-documented [30], making it easy to add new algorithms or dataset converters.

III. Methodology

The authors of the TreeCNN model [19] tested it using the Dataport [21] dataset and compared its performance against RNN and traditional CNN models. To explore whether this model can generalize across other datasets and how it compares to benchmark algorithms such as CO and Seq2Point, we integrated the TreeCNN model into the NILMTK toolkit. NILMTK is an open-source platform with comprehensive

documentation [30] for integrating
tion algorithms. Integrating the Tr
NILMTK toolkit consists of two key
the NILMTK-DF, and 2) Creating th
Fig. 1 illustrates the TreeCNN mode
the NILMTK toolkit.

A. Preprocessing the Data

Publicly available datasets vary in
NILMTK-DF is a standardized dat
the REDD dataset format. Existing d
into this standard format for use v
Once imported, NILMTK retains the
NILMTK-DF data structure. NILMT
data, relevant metadata, and other s
water, and temperature. This hierarc
tained as a Pandas DataFrame, index
columns for physically measured qua
for each appliance, mains (from the

The TreeCNN model works with
to its key properties: sparsity and re
temporal dimensions, significant varia
tion magnitudes across different appl
of unknown consumption sources [19]. Since the TreeCNN model is designed for high-dimensional hourly time series data ([Number of houses * Appliance No * Number of Days * Hours in a Day]), preprocessing is necessary before training the model. We defined a preprocessing function that takes Pandas DataFrame data and converts it to hourly data. Once the data conversion is complete, we feed the data to the TreeCNN model for training. This allows the TreeCNN model to work with any dataset available in the NILMTK toolkit.

B. Model Inclusion

NILMTK's disaggregation algorithms are located in the 'nilmtk/nilmtk/disaggregate/' directory and are implemented as Python classes. We created a class 'TreeCNN' that extends the Disaggregator superclass and implements all required methods. In this class, we used the 'partial_fit' (training method) and 'disaggregate_chunk' methods. In the '__init__' function, we set 'self.MODEL_NAME' to 'TreeCNN', which describes our algorithm name. The 'TreeCNN' class is then imported in the '__init__.py' file located within the 'nilmtk/nilmtk/disaggregate/' directory. Fig. 2 shows the TreeCNN model class algorithm in NILMTK toolkit.

The TreeCNN algorithm needs to learn how appliances consume energy from existing data. Disaggregation algorithms typically require appliance-level data from the building to be disaggregated (supervised) or from other buildings (unsupervised). The 'train_jointly' method in the 'api.py' class receives a MeterGroup data object containing a list of ElecMeter objects. This function extracts the Pandas DataFrame time series data of aggregated power and each appliance's power from the MeterGroup data. The mains and submeter data are then sent to the 'partial_fit' function of the TreeCNN model class for training. The trained model is stored in volatile memory and

```python
import torch

class TreeCNN(Disaggregator):
  def __init__(self, params):
    self.MODEL_NAME = 'TreeCNN'
    self.save_model_path = params.get('save-model-path', None)
    self.load_model_path = params.get('pretrained-model-path',None)

  def partial_fit(self, train_main, train_appliances, do_preprocessing=True,
  **load_kwargs):
    hd_data = self._preprocess_data(train_main, train_appliances)
    self.train_model(hd_data)

  def disaggregate_chunk(self, mains):
    hd_data = self._preprocess_data(mains)
    test_prediction_list = self.test_model(hd_data)
    return test_prediction_list

  def _preprocess_data(self, mains, appliances = None):
    #converts to one dimensional time series data to hourly high
dimensional  data
    ...
    return hd_data

  def train_model(self, hd_data):
    #train the TreeCNN model with hd_data
    ...

  def test_model(self, hd_data):
    test_prediction_list = []
    #test the TreeCNN model with hd_data
    ...
    return test_prediction_list
```

Fig. 2: TreeCNN model Python class in NILMTK toolkit.

will be lost once the disaggregator object is destroyed. The 'import_model()' and 'export_model()' methods can be used to create persistent models, allowing a model to be loaded from or saved to disk, facilitating incremental training on large datasets and sharing of pretrained models within the research community.

The trained TreeCNN model uses the 'disaggregate_chunk' method to generate predictions for each appliance. This method returns predictions as a Pandas DataFrame, with columns representing individual appliances and rows corresponding to time instants. The returned DataFrame's indexes must precisely match those of the input parameter DataFrame. Subsequently, 'api.py' class utilizes this prediction data from 'disaggregate_chunk' method to compute various performance metrics.

IV. Evaluation

A. Dataset Selection

We selected five publicly available datasets—REDD, UK-DALE, DRED, Smart*, and IDEAL—for training and evaluating NILM models. Table I characterizes these low-frequency datasets, all of which are measured in residential setups.

The Reference Energy Disaggregation Data Set (REDD) [22], was first published in 2011, contains both aggregated and sub-metered data from six households over several weeks. The UK-DALE dataset [23], published by the UK recording Domestic Appliance-Level Electricity project, comprises data from five households collected from 2012 to 2017, including both aggregate demand and ground truth demand for each appliance.

The Dutch Residential Energy Dataset (DRED) [31] from the Netherlands records both aggregated and appliance-level energy consumption from one household over a couple of months, with data collected every second. The Smart* dataset [32], published in 2012, provides a variety of environmental and operational data from three homes, measuring appliance power and total consumption in kilowatts. The IDEAL data corpus [33], published by the University of Edinburgh's School of Informatics, includes data from 255 UK homes over a 23-month period, with 39 of these homes containing both aggregated and sub-metered appliance data.

B. Appliances Selection

For our experiments, we considered the fridge, dishwasher, washing machine, microwave, and kettle appliances from the UK-DALE dataset. The REDD and Smart* datasets lack kettle data, while the DRED dataset includes only one building with fridge, washing machine, and microwave data. For the IDEAL dataset, we have taken fridge, dishwasher and washing machine appliances. These appliances significantly impact total power consumption and represent a diverse range of device types. The fridge remains constantly ON, whereas the dishwasher, washing machine, and similar appliances switch ON and OFF. Low sampling rates pose challenges for ON/OFF appliances because they may only be used briefly within an hour, which is a key reason why existing NILM algorithms often struggle with low sampling rate data.

C. Evaluation Metrics

Energy disaggregation research covers a wide range of application area. To meet this need, NILMTK [24] offers a collection of metrics that encompass both general detection metrics and those specific to energy disaggregation. For our research work, we have considered MAE [16], [24] and RMSE [24], [29] to measure and compare the performance of different energy disaggregation model. MAE and RMSE measure the precision of energy disaggregation, with RMSE being more sensitive to outliers.

Mean Absolute Error: It is defined by the sum of the differences between the predicted power $\hat{x}_i^{(n)}$ and the actual power $x_i^{(n)}$ of appliance n in each time slice t, divided by the appliance's total energy $x_t^{(n)}$ consumption (2).

$$\frac{\sum_t |x_i^{(n)} - \hat{x}_i^{(n)}|}{\sum_t x_t^{(n)}} \tag{2}$$

Root Mean Square Error: It is calculated between the predicted power $\hat{x}_i^{(n)}$ and the actual power $x_i^{(n)}$ of appliance n observed over T times (3).

$$\sqrt{\frac{1}{T}\sum_t (x_i^{(n)} - \hat{x}_i^{(n)})^2} \tag{3}$$

Lower MAE and RMSE values indicate better performance.

D. Experimental Setup

We compared the TreeCNN model with other baseline algorithms, including CO, FHMM, and Seq2Point, measuring MAE and RMSE parameters. For the TreeCNN model, we used 3000 epochs, while Seq2Point was run for 50 epochs with a batch size of 64. Details of the hardware and software configurations are provided in Table II. The TreeCNN model works with hourly data. And, the TreeCNN model constructs a tree structure using different appliances. Finding the optimal tree structure is an NP-complete problem [19]. In our experiments, we used four different orders of appliances to form the tree structure for the model. From these, we identified the optimal tree structure to plot the TreeCNN graph. Additionally, we averaged the energy disaggregation results from all four tree structures to create the TreeCNN_Avg plot. Table III provides details of the training and testing building data for each dataset. It also includes the optimal appliance order for TreeCNN, which was chosen from the four possible orders used in our experimental setup.

E. Experimental Results

Our evaluation spans three key scenarios: (1) same building training and testing, (2) cross-building evaluation within the same dataset, and (3) cross-dataset generalization. While traditional mathematical models (CO and FHMM) generally underperformed across all scenarios, we focus our analysis on comparing the deep learning approaches - TreeCNN and Seq2Point. We will use the following abbreviations for appliances throughout: Fridge (F), Dishwasher (DW), Washing Machine (WM), Microwave (MW), and Kettle (K).

In the DRED dataset (Case 1), as shown in Fig. 3a, TreeCNN demonstrated superior performance for ON/OFF devices, with notably low MAE values of 3.43 and 7.89 for WM and MW respectively. The model's CNN filters effectively captured both weekly patterns (WM) and daily patterns (MW), despite the potential overshadowing effect of high-power appliances. Fig. 3b further highlights TreeCNN's superior prediction performance with RMSE values of 52.11 and 45.02 for WM and MW respectively. This performance demonstrates TreeCNN's ability to handle appliances with different temporal patterns simultaneously.

For the Smart* dataset (Case 1, Building 3), we split data into training (2014-01-01 to 2014-06-30) and testing (2014-07-01 to 2014-08-30) periods. As shown in Fig. 4a, Seq2Point achieved the lowest MAE (0.052) for the F, while TreeCNN showed consistent performance across all appliances, though with 26% higher MAE for DW compared to baseline models. The RMSE values (Fig. 4b) for DW, WM, and MW are almost identical between the models, indicating that both deep learning approaches handle power variation similarly for these appliances.

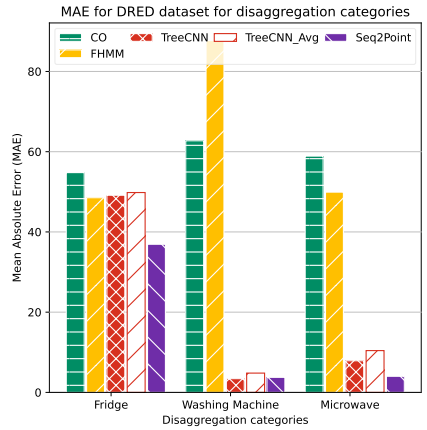
The REDD dataset evaluation (Case 2) focused on cross-building generalization, with training on buildings 2, 3, 5 and testing on building 1. During the period from 2011-04-01 to 2011-05-30, WM power consumption in buildings 2 and 5 ranged between 0-6 Watt, indicating infrequent use. Fig.

TABLE I: Dataset Information

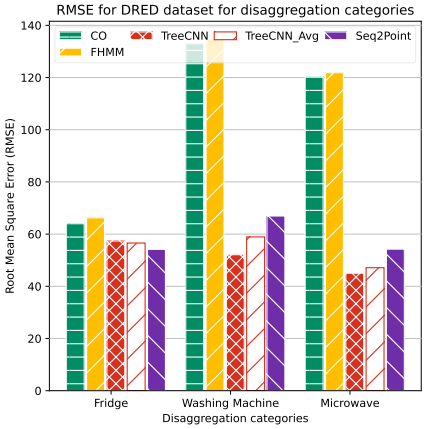
Dataset	Number of Buildings	Institution	Location	Appliance Sample Frequency	Aggregate Sample Frequency
REDD	6	MIT	Massachusetts, US	3s	1s
UK-DALE	5	Imperial College	London, UK	6s	1-6s
DRED	1	TUDelft	Netherlands	1s	1s
Smart*	3	UMass	Western Massachusetts, US	1s	1s
IDEAL	255	University of Edinburgh	UK	5s	1s

TABLE II: System Configurations

OS	Ubuntu 22.04 LTS
CPU	2x Xeon Silver 4314 (16c, 32t)
GPU	Nvidia A40
Memory	192 GB
Storage	4x 960GB NVMe SSD
Python	3.8.17
PyTorch	2.0.0



(a) MAE measurement



(b) RMSE measurement

Fig. 3: MAE & RMSE measurements of DRED dataset for Case 1 (Building 1 data is splitted into training (July 05, 2015, to October 30, 2015) and testing (November 01, 2015, to December 05, 2015) periods).

5a shows both CNN-based models significantly outperformed traditional approaches, with Seq2Point showing exceptional performance across all four appliances, achieving a 56.96% reduction in MAE for F compared to FHMM. This superior performance stems from better handling of varying power consumption patterns between buildings. The RMSE results (Fig. 5b) further highlight the strong performance of TreeCNN and Seq2Point. Additionally, we verified MAE for FHMM and Seq2Point algorithms for F, DW appliances with the mentioned training and testing setup from [34], which validates our implementation's consistency.

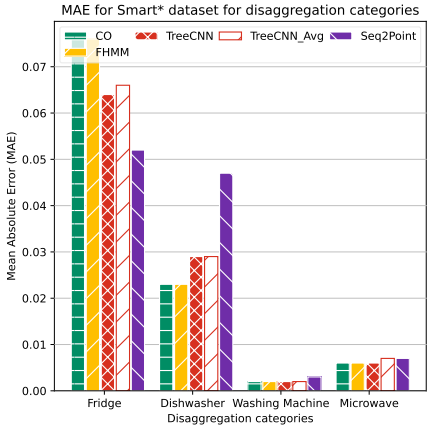
In the UK-DALE dataset (Case 2), Fig. 6a demonstrates that TreeCNN maintained consistent performance with MAE ranging from 34 to 46 across all five appliances, while Seq2Point achieved the lowest MAE ($\approx$7) for WM. As shown in Fig. 6b, both models handled sharp power usage peaks effectively, particularly evident in kettle disaggregation where traditional approaches struggled with short, intense bursts of activity. Our measured MAE aligned for Seq2Point for F, DW with the reported training and testing setup from the paper [34].

The IDEAL dataset (Case 2) evaluation (Fig. 7a) highlighted TreeCNN's strength in handling WM (MAE $\approx$64) but revealed limitations with DW (MAE $>$90) due to diverse usage patterns. We observed that power consumption range of WM in testing unseen data differs from training data, with sharp consumption peaks present in the dataset. TreeCNN outperformed Seq2Point by 2.8x for WM, suggesting better handling of operational cycles. Fig. 7b demonstrates TreeCNN's superior performance for F and WM appliances, though it struggles with DW (RMSE $\approx$350).

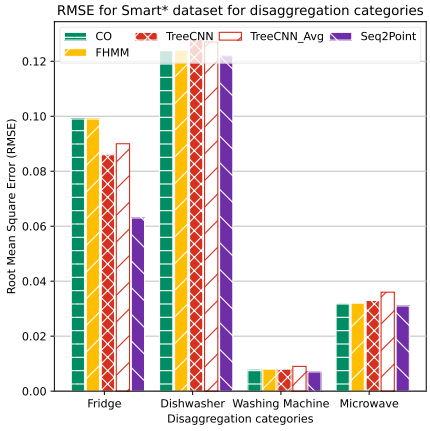
Finally, in our cross-dataset experiment (Case 3) using UK-DALE for training and REDD for testing, Fig. 8a shows both CNN models demonstrated superior generalization capabilities. TreeCNN excelled in MW prediction (MAE $<$25) but struggled with DW (MAE $\approx$80) due to varying power consumption patterns between the UK-DALE and REDD datasets. The hierarchical tree structure of TreeCNN shows

TABLE III: Training & Testing Information about Datasets

Dataset	Trained on Building Data	Tested on Building Data	Appliances Order for TreeCNN
DRED	1	1	F, WM, M
Smart*	3	3	F, DW, WM, MW
REDD	2, 3, 5	1	F, WM, DW, MW
UK-DALE	1, 5	2	F, DW, WM, MW, K
IDEAL	136, 175	105	F, WM, DW
UK-DALE & REDD	1, 2 (UK-DALE)	1 (REDD)	F, WM, DW, MW

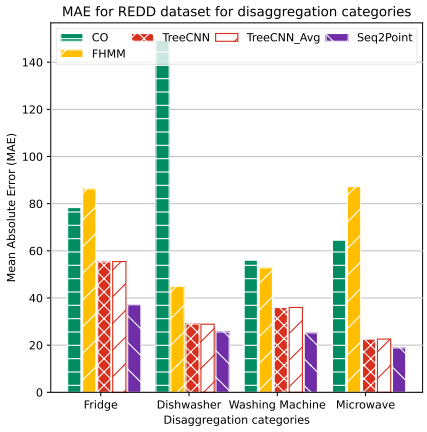


(a) MAE measurement

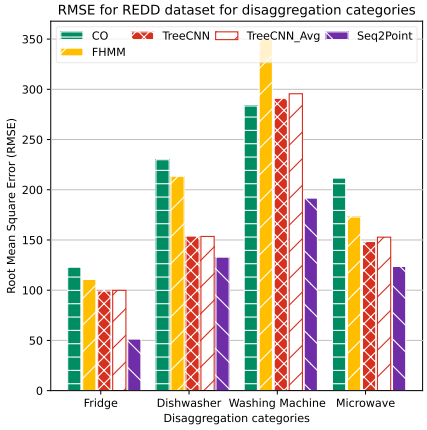


(b) RMSE measurement

Fig. 4: Smart* dataset's MAE & RMSE measurements for Case 1 (The power measurement for this dataset is in kilowatts, so the MAE and RMSE values for all models are below 1).



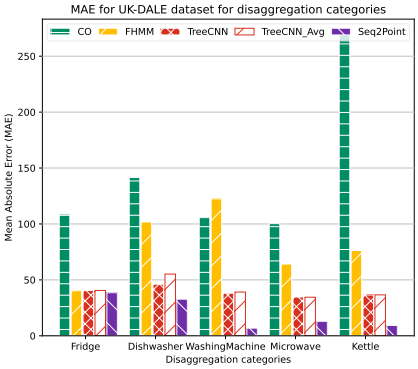
(a) MAE measurement



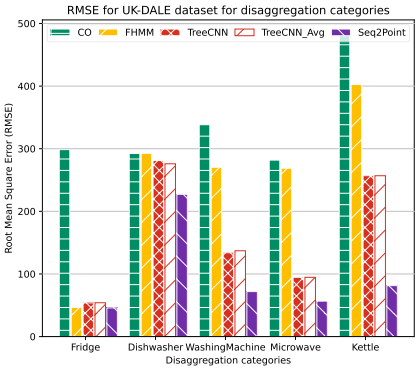
(b) RMSE measurement

Fig. 5: REDD dataset's MAE & RMSE results for appliances F, DW, WM and MW for Case 2.

limitations in capturing diverse DW patterns efficiently. Fig. 8b illustrates Seq2Point's consistent performance across all appliances, attributed to its sequence-to-point learning approach that better handles complex temporal sequences. Additionally, our measured MAE for F, WM appliances for CO, FHMM, and Seq2Point are consistent with the paper [35] findings.

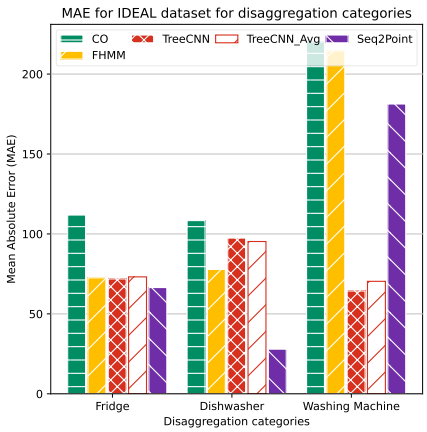


(a) MAE measurement

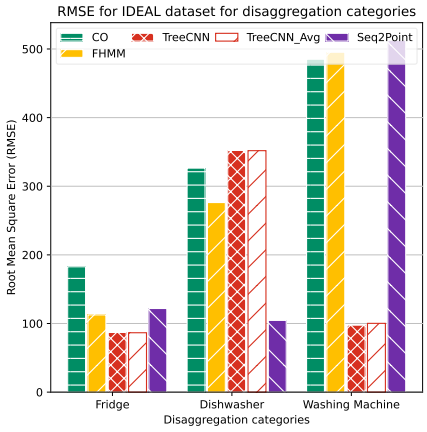


(b) RMSE measurement

Fig. 6: UK-DALE dataset experiment for five household appliances F, DW, WM, MW and K for Case 2.



(a) MAE measurement

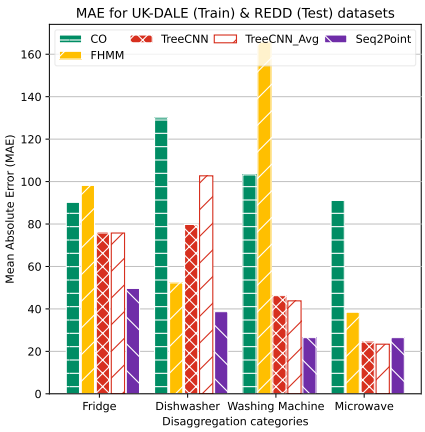


(b) RMSE measurement

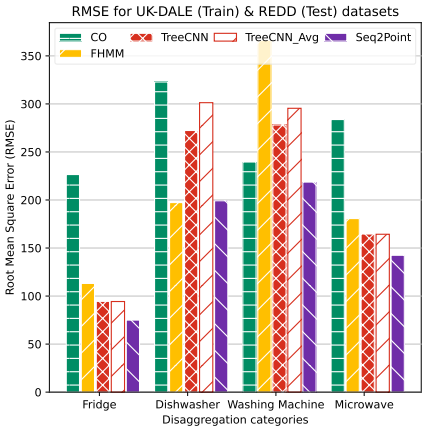
Fig. 7: MAE & RMSE values of appliances F, DW and WM for the IDEAL dataset under Case 2 scenario.

F. Inference Time Measurement

Trained NILM models are deployed in systems to provide predictions for unseen household appliance data. The time it takes for a trained model to generate a prediction for a sample is called inference time, which is critical because users expect rapid forecasts that can handle many requests efficiently. To better understand this metric, we measured the serving time for one sample using the Smart* dataset, focusing on the fridge appliance, which is common in every household. We trained four algorithms—CO, FHMM, Seq2Point, and TreeCNN—using data from the building 1, covering the period from 2014-07-01 to 2014-09-30. We recorded the inference time for 100 samples and averaged the results in milliseconds. Table IV shows the measured inference times for the four models. Both CO and FHMM models took around 47 milliseconds per sample. FHMM uses a probabilistic model with fixed hidden states and transition probabilities, while CO involves solving an optimization problem based on precomputed patterns. Both of these models employ relatively straightforward mathematical operations for inference. These factors contribute to their short inference times. Seq2Point model took an average of 120 milliseconds per sample. As a DNN model with many CNN layers, Seq2Point's inference process involves passing data through multiple layers of neurons, requiring numerous matrix multiplications and nonlinear transformations. This complexity results in a longer inference time compared to simpler models like CO and FHMM. TreeCNN model took approximately 70 milliseconds per sample. TreeCNN uses fewer CNN layers than Seq2Point and employs a hierarchical tree structure to efficiently capture temporal patterns in appliances. This balance between capturing detailed temporal patterns and maintaining efficiency results in a medium inference time, longer than CO and FHMM but shorter than Seq2Point. These results highlight the trade-offs between model complexity and inference time, with simpler models being faster but potentially less accurate, and more complex models being slower but more capable of capturing intricate patterns in the data.



(a) MAE measurement



(b) RMSE measurement

Fig. 8: MAE & RMSE values for appliances F, DW, WM and MW for Case 3 (Trained with the UK-DALE dataset and tested with the REDD dataset).

TABLE IV: Inference Time for NILM Models

Model Name	Inference Time (ms)
CO	47.8
FHMM	47.92
Seq2Point	120.03
TreeCNN	70.76

V. Discussion

A. Limitations, Insights and Scope for Improvement

Our comprehensive evaluation of TreeCNN across multiple datasets reveals several key limitations and opportunities for enhancement. First, while TreeCNN excels in single-building scenarios, its performance degrades when generalizing across different datasets, particularly for appliances with varying power consumption patterns. This limitation stems from the model's rigid tree structure, which, once optimized for a specific building's temporal patterns, may become suboptimal when applied to significantly different usage contexts. The model particularly struggles with appliances exhibiting sharp power consumption peaks (e.g., dishwashers) or highly variable usage patterns across households. This challenge is compounded by the current greedy approach for determining tree structure, which may not achieve global optimality. To address these limitations, several promising directions for improvement emerge: (1) implementing an adaptive tree structure that can dynamically adjust based on observed patterns and real-time performance metrics, (2) incorporating attention mechanisms to better handle varying appliance importance and temporal patterns, and (3) developing transfer learning capabilities to improve cross-dataset generalization. Additionally, the integration of contextual features such as time of day, seasonal patterns, and geographical factors could enhance the model's ability to capture more complex usage patterns. The current implementation's assumption of consistent temporal patterns may not hold across different cultural or geographical contexts, suggesting the need for more sophisticated pattern recognition mechanisms.

B. Comparative Analysis with Seq2Point

Our experimental results demonstrate fundamental trade-offs between TreeCNN and Seq2Point models, rooted in their architectural differences and reflected in their performance characteristics. While TreeCNN achieves superior performance in single-building scenarios through its recursive decomposition approach, Seq2Point demonstrates more robust generalization across different datasets and buildings. This difference stems from their core architectural designs: Seq2Point excels at capturing localized temporal coherence through its sliding window mechanism, enabling accurate predictions for ON/OFF appliances with sharp and infrequent power consumption peaks. Its sequence-to-point learning approach, which predicts a target appliance's power consumption for a specific point within a window, effectively reduces error accumulation that can plague hierarchical models. TreeCNN, conversely, employs a hierarchical decomposition strategy that recursively reduces aggregate power readings into appliance-specific patterns. While this approach is particularly effective for continuous appliances like refrigerators and scenarios with regular, well-separated appliance patterns, it can struggle with appliances exhibiting sporadic usage patterns due to residual propagation through the tree structure. This architectural distinction manifests in computational differences as well - Seq2Point requires approximately 120 milliseconds for inference compared to TreeCNN's 70.76 milliseconds. Though this 50 milliseconds difference may seem minimal for offline applications, it becomes significant in large-scale deployments processing millions of data points. The choice between these models thus depends on specific application requirements: TreeCNN is more suitable for single-building deployments requiring faster inference and handling stable usage patterns, while Seq2Point's robust error handling and temporal coherence make it better suited for multi-building deployments and cross-regional applications where consistent

generalization is paramount. Future research could potentially bridge this gap by developing hybrid approaches that combine TreeCNN's hierarchical efficiency with Seq2Point's robust temporal pattern recognition capabilities.

VI. Conclusion

In this paper, we integrated the TreeCNN model into the NILMTK toolkit and evaluated its performance against baseline models across diverse datasets. TreeCNN excelled in single-building scenarios by effectively isolating appliance patterns with its hierarchical decomposition, achieving low MAE and RMSE for both ON/OFF and continuous appliances. However, its performance diminished in cross-building and cross-dataset evaluations, particularly for irregular or variable appliance usage patterns like dishwashers. Comparatively, Seq2Point demonstrated better generalization across datasets due to its sequence-to-point architecture, albeit with a slightly higher inference time (120ms vs. 70ms). Deep learning models like TreeCNN and Seq2Point outperform traditional approaches in NILM, but TreeCNN's limitations—rigid tree structure, error propagation, and suboptimal generalization—highlight opportunities for improvement. Future work will focus on adaptive tree structures, transfer learning for cross-dataset generalization, and pretraining on unlabeled data to enhance its performance.

Acknowledgment

We thank the anonymous reviewers for their insightful comments and feedback. This work has been partially funded by NSF grants CSR-2106634, CSR-2312785, CCF-1919113, and OAC-2004751, along with UKRI-EPSRC grant EP/X035085/1.

References

[1] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," in *Energy and buildings 40, no. 3*, 2008, pp. 394–398.

[2] S. S. Sharma, "Determinants of carbon dioxide emissions: empirical evidence from 69 countries," in *Applied energy 88, no. 1*, 2011, pp. 376–382.

[3] "Energy star: Save energy," https://www.energystar.gov/buildings.

[4] E. Aydin, D. Brounen, and N. Kok, "Information provision and energy consumption: Evidence from a field experiment," in *Energy Economics 71*, 2018, pp. 403–410.

[5] T. N. et al., "Recent Challenges and Methodologies in Smart Grid Demand Side Management: State-of-the-Art Literature Review," in *Mathematical Problems in Engineering 2021, no. 1*, 2021, p. 5821301.

[6] Y. W. et al., "Review of smart meter data analytics: Applications, methodologies, and challenges," in *IEEE Transactions on Smart Grid 10, no. 3*, 2018, pp. 3125–3148.

[7] S. C. et al., "Nonintrusive load monitoring based on self-supervised learning," in *IEEE Transactions on Instrumentation and Measurement 72*, 2023, pp. 1–13.

[8] L. Yin and C. Ma, "Interpretable Incremental Voltage-Current Representation Attention Convolution Neural Network for Nonintrusive Load Monitoring," in *IEEE Transactions on Industrial Informatics 19, no. 12*, 2023, pp. 11 776–11 787.

[9] G. W. Hart, "Nonintrusive appliance load data acquisition method: Progress report," in *MIT Energy Laboratory*, 1984.

[10] G. W. Hart, "Nonintrusive appliance load monitoring," in *Proceedings of the IEEE 80, no. 12*, 1992, p. 1870–1891.

[11] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," in *Artificial intelligence and statistics*, 2012, pp. 1472–1482.

[12] L. Mauch and B. Yang, "A novel DNN-HMM-based approach for extracting single loads from aggregate power signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2384–2388.

[13] J. H. et al., "MSDC: exploiting multi-state power consumption in non-intrusive load monitoring based on a dual-CNN model," in *Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 4*, 2023, pp. 5078–5086.

[14] J. C. et al., "Temporal and spectral feature learning with two-stream convolutional neural networks for appliance recognition in NILM," in *IEEE Transactions on Smart Grid 13, no. 1*, 2021, pp. 762–772.

[15] L. Mauch and B. Yang, "A new approach for supervised power disaggregation by using a deep recurrent LSTM network," in *IEEE global conference on signal and information processing (GlobalSIP)*, 2015, pp. 63–67.

[16] C. Z. et al., "Sequence-to-point learning with neural networks for non-intrusive load monitoring," in *Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1*, 2018.

[17] B. G. et al., "Deep learning based non-intrusive load monitoring for a three-phase system," in *IEEE Access 11*, 2023, pp. 49 337–49 349.

[18] "Staff Report. 2017," in *2017 Assessment of Demand Response and Advanced Metering*.

[19] Y. Jia, N. Batra, H. Wang, and K. Whitehouse, "A tree-structured neural network model for household energy breakdown," in *The World Wide Web Conference*, 2019, pp. 2872–2878.

[20] J. Kelly and W. Knottenbelt, "Neural NILM: Deep neural networks applied to energy disaggregation," in *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*, 2015, pp. 55–64.

[21] O. P. et al., "Dataport and NILMTK: A building data set designed for non-intrusive load monitoring," in *IEEE global conference on signal and information processing (GlobalSIP)*, 2015, pp. 210–214.

[22] J. Z. Kolter and M. J. Johnson, "REDD: A public data set for energy disaggregation research," in *Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA, vol. 25, no. Citeseer*, 2011, pp. 59–62.

[23] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," in *Scientific data 2, no. 1*, 2015, pp. 1–14.

[24] N. B. et al., "NILMTK: An open source toolkit for non-intrusive load monitoring," in *Proceedings of the 5th International Conference on Future Energy Systems*, 2014, pp. 265–276.

[25] W. L. et al., "Industrial load disaggregation based on Hidden Markov Models," in *Electric Power Systems Research 210*, 2022, p. 108086.

[26] W. K. et al., "A hierarchical hidden markov model framework for home appliance modeling," in *IEEE Transactions on Smart Grid 9, no. 4*, 2016, pp. 3079–3090.

[27] Z. Ghahramani and M. I. Jordan, "Factorial hidden markov models machine learning," in *Kluwer Academic Publishers*, 1997.

[28] P. H. et al., "Review on deep neural networks applied to low-frequency NILM," in *Energies 14, no. 9*, 2021, p. 2390.

[29] Z. S. et al., "Multiscale self-attention architecture in temporal neural network for nonintrusive load monitoring," in *IEEE Transactions on Instrumentation and Measurement 72*, 2023, pp. 1–12.

[30] "NILMTK documentation," https://github.com/nilmtk/nilmtk/tree/master/docs/manual.

[31] A. S. U. Nambi, A. R. Lua, and V. R. Prasad, "LocED: Location-aware energy disaggregation framework," in *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, 2015, pp. 45–54.

[32] S. B. et al., "Smart*: An open data set and tools for enabling research in sustainable homes," in *SustKDD, August 111, no. 112*, 2012, p. 108.

[33] M. P. et al., "The IDEAL household energy dataset, electricity, gas, contextual sensor data and survey data for 255 UK homes," in *Scientific Data 8, no. 1*, 2021, p. 146.

[34] J. H. et al., "InFocus: Amplifying critical feature influence on non-intrusive load monitoring through self-attention mechanisms," in *IEEE Transactions on Smart Grid 14, no. 5*, 2023, pp. 3828–3840.

[35] N. B. et al., "Towards reproducible state-of-the-art energy disaggregation," in *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, 2019, pp. 193–202.