A Comparative Study of Event Detection Algorithms for Energy Disaggregation

RUBEN DE KONING, University of Twente, The Netherlands

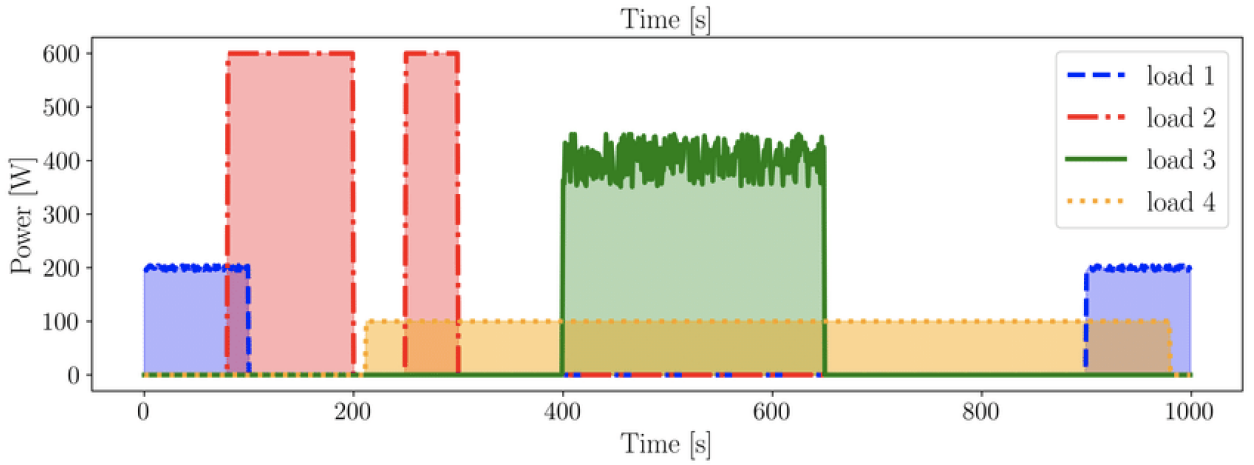

Fig. 1. An example of a disaggregated energy consumption graph; the result of an energy disaggregation framework. Source: [1]

This research focuses on different event detection algorithms used in Non-Intrusive Load Monitoring (NILM) systems for energy disaggregation. The study will evaluate these algorithms and compare them based on their accuracy, precision, recall, f1-score, and computational complexity. The primary goal is to determine a suitable algorithm for residential energy disaggregation using real-world data. Such an algorithm could be incorporated into an energy disaggregation framework, which generates a disaggregated energy consumption graph similar to the one shown in Figure 1.

Additional Key Words and Phrases: NILM, Non-Intrusive Load Monitoring, Energy Disaggregation, Data-driven algorithms, computational complexity

1 INTRODUCTION

In recent years, there have been two significant trends in the energy sector: the increase in locally generated energy by consumers and a surge in overall energy consumption. Both trends highlight the need for consumers to gain insights into their energy consumption and to help them align their energy consumption with their energy production to maximize the use of locally generated energy.

As depicted in Figure 2, there are two ways to derive the household's total energy consumption from individual appliance usage: the intrusive and the non-intrusive way. The intrusive method directly measures each appliance's energy usage, whereas Non-Intrusive Load Monitoring (NILM) leverages data from a single measurement point and uses advanced algorithms to deduce appliance-level usage patterns. This usage of this NILM technique has grown significantly in recent years due to its cost-effectiveness, scalability, and ability to ensure consumer privacy. However, NILM faces computational complexity challenges, as it often relies on high-end hardware, which limits the widespread adaption of this technique. It is

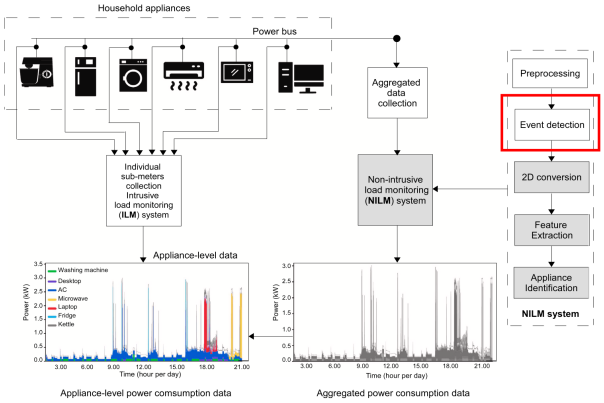

Fig. 2. A schematic overview of two approaches to create a disaggregated energy consumption graph: ILM and NILM. Highlighted is the event detection phase within a NILM system. Source: [2]

TScIT 40, February 2, 2024, Enschede, The Netherlands

therefore crucial to address to issue to make non-intrusive energy disaggregation accessible to everyone. In addition to the example framework in Figure 2, it is important to integrate a validation step into the NILM system. This additional step ensures the accuracy and reliability of the system's outcome and without this validation, any conclusions or discussions based on these results would be fundamentally flawed.

This study fits into a niche that uses unsupervised learning techniques for event detection in time-series data, evaluated using supervised learning metrics, which puts it in a unique position. In classification, an algorithm categorizes data into predefined classes. Similarly, event detection algorithms classify data sequences into 'event' and 'non-event' categories. The key difference is that event detection specifically identifies significant occurrences within data streams. This approach contributes to the broad field of Machine Learning (ML), particularly in the areas of unsupervised anomaly detection, time-series analysis, and signal processing.

The evaluation consists of four event detection algorithms: Local Threshold, Adaptive Threshold, K-means clustering, and Derivate-based. Their performance is assessed using accuracy, precision, recall, and F1-score metrics. During the evaluation, a private dataset of Saxion University of Applied Sciences is used, which has been manually labeled to serve as ground truth and a median filter is applied before usage. Before the evaluation, the algorithms undergo validation using a single appliance signal of the UK-Dale dataset[3].

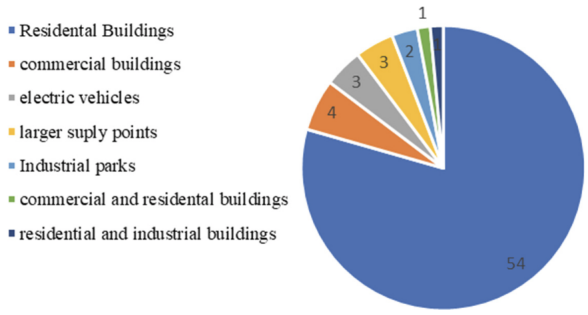

Fig. 3. Research domain. Source: [8]

Metrics	*k*-means	Peak_Detection	LTED
TP	1424	1396	1450
TN	401782	401254	401717
FP	66	594	131
FN	395	649	**298**
Precision	95.5	70.1	**95**
Recall	78.2	68.2	**82**
f1-score	86.06	69.1	**88**
TPR	0.78	0.68	**0.82**
FPR	0.01	0.14	0.03
FNR	0.21	0.31	**0.17**

Fig. 4. Comparison of three event detection algorithms: k-means clustering, peak detection, and local threshold-based. Source: [9]

2 RELATED WORK

Energy disaggregation has become an increasingly popular research area, and the development and evaluation of event detection algorithms for energy disaggregation frameworks have followed accordingly. Hart [4] initiated this field in 1992, aiming to non-intrusively separate energy consumption into individual appliance-level components. Approaches like Hidden Markov Models (HMM) [5] demonstrated their effectiveness in capturing temporal dependencies within energy consumption data about a decade ago, while optimization-based methods [6] addressed the problem as a combinatorial optimization task at about the same time. More recently, neural network-based solutions [7] have gained attention for their ability to learn complex patterns. As can be read in Figure 3, current research primarily focuses on residential energy disaggregation. Tolnai et al. [8] also address correlated challenges, such as the limited computational capacity of smart meters, which hinders algorithm integration—highlighting the significance of taking computational complexity into account when evaluating algorithms used for energy disaggregation frameworks.

The selection of four specific event detection algorithms for the study, namely the Adaptive Threshold, Derivate, Local Threshold, and K-Means clustering-based approaches, is based on an evaluation of their performance and suitability in the context of energy disaggregation. Islam and Shah's research [9] showcases a comparison of event detection algorithms, namely Peak Value, K-Means Clustering, and Local Threshold-based approaches. Their findings show that the peak value-based algorithm underperformed compared to the other two as can be seen in Figure 4, hence the K-Means Clustering and Local Threshold-based methods are chosen for this study. This study considers two more algorithms: an Adaptive Threshold algorithm based on standard deviation, presented in a recent paper [10] published in February 2023, and a derivative-based algorithm [11] with promising results.

All algorithms have their own distinct way of detecting events. For the Local Threshold-based algorithm as shown in Appendix B, a windowing technique is used in which data is divided into windows of a specific length, and statistical parameters such as the mean and standard deviation are computed for each window to provide information about the overall behavior and variability of the signal within each window. Additionally, the mean and standard deviation of all windows are computed to help comprehend the distribution of power fluctuations of the overall signal. However, the threshold is mostly dependent

on the mean of that window. The Adaptive Threshold-based algorithm as shown in Appendix C works in a similar way as it also makes use of the windowing technique and of the mean and standard deviation. However, the most significant difference lies in how the threshold is calculated. The Adaptive Threshold-based algorithm uses a more complex formula involving the window size and the ratio of maximum to mean power, whereas the Local Threshold-based algorithm uses a simpler linear combination of mean and standard deviation.

The K-Means Clustering-based algorithm as shown in Appendix A first computes the optimal number of clusters using the Silhouette method, this is a very computationally complex task and therefore the algorithm only looks for the optimal number between 2 and 7 clusters. After each measurement is assigned to a specific cluster, the algorithm iterates over all measurements and an event is detected when the previous measurement is a different cluster than the current one. On the other hand, the Derivative-based algorithm as shown in Appendix D consists of three parts: preprocessing, changes-detection, and peak-detection. During the preprocessing phase, the aggregate power signal is altered to the current signal envelope to avoid the algorithm being vulnerable to voltage drops, as the electrical current is the immediate cause of appliance activities and power also includes voltage. During the changes-detection phase, a derivation is performed to only keep the changes in the signal and filter out the steady states. Next, the derived signal is squared to increase the gap between small changes due to noise, and significant changes due to transient states. After that, the peaks are detected and only the significant peaks are seen as events.

3 METHODOLOGY

During this research, our methodology is similar to the CRISP-DM methodology and will consist of the following steps:

(1) `Data understanding`: Understand the data format that will be used.
(2) `Dataset preparation`: Prepare various datasets to ensure the dataset spans different periods, household profiles, and appliance usage scenarios to capture real-world dynamics.
(3) `Implementation`: Implementation of the algorithms.
(4) `Ground truth annotation`: Manually set the ground truth for a subset of the data to serve as a benchmark. This is important for validating the performance of each algorithm.
(5) `Validation`: Validate the implementation of the algorithms by using the UK-Dale single appliance data. In case of incorrect results, return to the implementation step.
(6) `Evaluation`: Evaluate the performance of the algorithms on accuracy, precision, recall, F1-Score, and computational complexity using the Saxion Data.

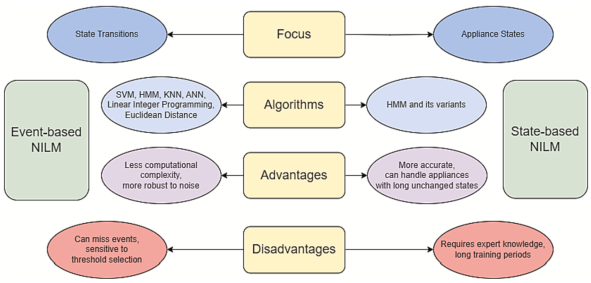

Fig. 5. Summary of the features of event-detection. Source: [12]

(7) `Visualization`: Presenting the results in informative formats to show a good overview of the results and understanding of each algorithm's strengths and weaknesses.

By adhering to the prescribed steps, a clear overview of the algorithm's performance can be obtained, which can then be used for a comprehensive discussion and as a foundation for future research.

Due to the unavailability of the REDD dataset by MIT at the time of this study, the publicly available UK-Dale dataset is used during the validation step. More specifically, a subset of the UK-Dale dataset that only measured the power signal of a single appliance, is used to validate the algorithms. The Saxion Data, which is not a publicly available dataset, was received in an unlabeled format. To enable an algorithmic evaluation, the data had to be labeled first. A 3-day interval of the Saxion Data is taken at random for testing the algorithms on an aggregate power signal. For a more comprehensive assessment, a 10-day interval of the Saxion Data was randomly selected, which is mutually exclusive to the 3-day interval. The algorithms were then evaluated based on the specified metrics and compared against each other.

The chosen algorithms can be categorized according to the summary of features in Figure 5, which illustrates two main types of event detection algorithms: state-based and event-based. The Derivate, Adaptive Threshold, and Local Threshold-based algorithms all align with the event-based category, as they focus on mathematical event detectors to identify events. In contrast, the K-Means Clustering algorithm falls into the state-based category, as it relies on state changes rather than event detectors to identify events. Additionally, a modification is made to the K-Means Clustering algorithm, instead of the proposed Elbow Method to identify the optimal number of clusters, the Silhouette method is used to be able to automatically evaluate the algorithm.

During the testing and evaluation phase, a standard set of metrics is used to assess the algorithms, consisting of accuracy, precision, recall, and f1-score. The formulas to compute

these metrics are listed below, including a description stating their relevance. To determine the suitability for real-time application, the time complexity is also included in the assessment and verified using a plot.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of measurements}} \tag{1}$$

Accuracy shows the overall correctness of an algorithm, but it may not be as informative on its own, especially if the data is imbalanced, where there are many more non-events that events.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

Precision measures an algorithm's ability to avoid false positives, important when the cost of falsely detecting an event is high.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

Recall is essential when missing an event is costly. It demonstrates the algorithm's capability to identify all actual events.

$$F1\text{-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The F1-Score balances the precision and recall, which is particularly valuable when dealing with an imbalanced dataset because it equally considers the algorithm's performance on the minority class, which are the events in this case.

4 RESULTS

As described in the methodology section, several steps are undertaken to conduct a comprehensive evaluation of the four event detection algorithms. Initially, the algorithms are validated using the UK-Dale dataset by Kelly [3], to make sure they perform as intended. The validation process can be seen as the most important step to lay a robust foundation for upcoming analyses.

Following the validation step, the time complexity of each algorithm is computed, and the algorithms are then applied to aggregate power signals. Firstly, parameter adjustments were made to make them suitable for this specific use-case using 3-day data, and then 10-day data to determine their accuracy, precision, recall, and f1-score. Allowing the discussion to be done on accurate data, retrieved in a precise and above all, reproducible manner.

4.1 Validation

The REDD dataset by MIT[13] was used in some of the papers to receive the promising results. However, at the time of this study, the REDD database is not available. So, to verify whether the algorithms indeed correctly detect events, the UK-Dale dataset is used. Specifically, the subset: House 3, appliance 2, between the first of March 2013 and the third of March 2013. This subset is chosen at random, and the power signal of this subset can be seen in Figure 6.

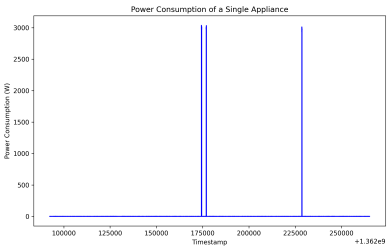

Fig. 6. The power signal of the single appliance of the UK-Dale dataset.

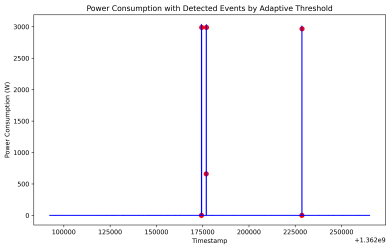

Fig. 7. The power signal of the single appliance of the UK-Dale dataset with events detected by the Adaptive Threshold algorithm.

The use of the single appliance signal is because it is easily verifiable that the device was indeed turned on or off when the algorithms detect an event. For context, according to the labeling done in the dataset, this specific appliance is officially turned off when the power consumption is below 5W. Anything below this threshold can be considered as standby consumption. The results of all four algorithms are identical and easily verifiable, as can be seen in Figure **??** showcasing the resulting graph of the Adaptive Threshold-based algorithm.

All four algorithms appear to detect an event at approximately 600W. However, this event is not an error but rather a consequence of the event detection methodology used. Events are marked right before the signal reaches a steady state. Consequently, when the transient state consists of a single measurement, the event is marked at the end of the preceding steady state, as this is the measurement right before the start of the next steady state. This specific occurrence is enlarged in Figure 8, where it becomes evident that an additional measurement exists between steady states by the difference in slope coefficient before and after the marker when the device is powered on, as opposed to when it is turned off.

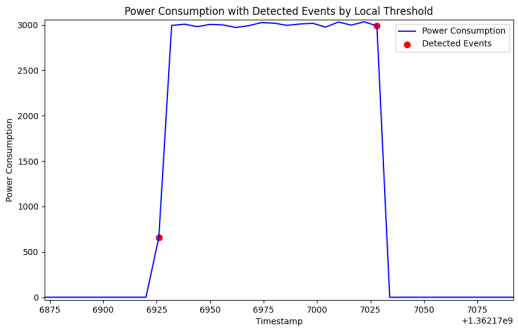

Fig. 8. Enlargement of the "faulty" event detected by the algorithms.

Table 1. Time Complexity of the Algorithms

Algorithm	Time Complexity
Local Threshold	$O(N)$
Adaptive Threshold	$O(N)$
Derivative	$O(N)$
K-Means Clustering	$O(N^2)$

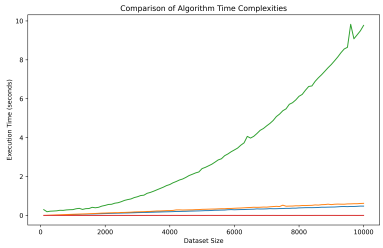

Fig. 9. Graph showing the time complexity of the algorithms.

4.2 Time complexity

As mentioned before, the time complexity of an algorithm plays a crucial role in allowing the algorithm to run in real-time at the consumer's house where there is a common absence of high-end hardware. The algorithms' time complexity is plotted in Figure 9 and noted in Table 1. As the Adaptive Threshold-based algorithm's implementation is making use of highly optimized functions the scale is different from the others. However, in Figure 10 we can see that the algorithm also has a linear time complexity.

4.3 Testing

During the testing phase, it is important to ensure that the algorithm's parameters are suitable for the evaluation data. It is striving for suitability rather than optimality to prevent overfitting. To achieve this, a multi-day testing approach is employed, as opposed to testing on a single day, and by ensuring the testing data includes a variety of appliances. This diversity

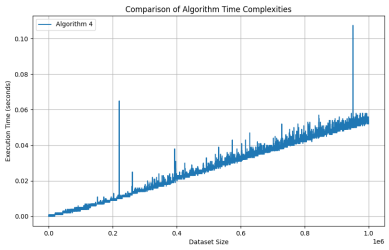

Fig. 10. Graph showing the time complexity of the Adaptive Threshold algorithm.

Table 2. Performance Metrics for Testing

Algorithm	Accuracy	Precision	Recall	F1 Score
Local Threshold	0.9997	0.9219	0.8433	0.8778
Adaptive Threshold	0.9992	0.7657	0.8744	0.7761
Derivative	0.9995	0.8795	0.7206	0.7905
K-Means Clustering	0.9962	0.4167	0.7287	0.4521

Table 3. Performance Metrics for Evaluation

Algorithm	Accuracy	Precision	Recall	F1 Score
Local Threshold	0.9996	0.8661	0.7966	0.8276
Adaptive Threshold	0.9996	0.7847	0.9149	0.8335
Derivative	0.9995	0.8231	0.7601	0.7780
K-Means Clustering	0.9980	0.4428	0.7101	0.5195

is to give confidence that the evaluation data is reasonably represented. The outcomes of the testing phase, conducted on the 3-day Saxion Data, are summarized in Table 2. This table provides the average values of the performance metrics mentioned before. The tables in Appendix E contain more detailed information on the three individual days.

4.4 Evaluation

During the evaluation phase, the algorithm's parameters determined in the testing phase are used and the algorithms are executed and assessed on ten days of Saxion Data to get a better understanding of their performance. This 10-day subset of the Saxion Data is mutually exclusive with the 3-day subset used in the testing phase. The results of the evaluation are summarized in Table 3. This table contains the average values of the performance metrics mentioned before. The tables in Appendix F contain more detailed results of the first 2 days of the 10-day subset.

5 DISCUSSION

Correctly detecting an event

In energy consumption monitoring, it is important for the detection of events to determine exactly when an event happens. This all starts with exactly pinpointing when an event starts

and ends and marking it at the right spot to be able to automatically use it in a later stage in the framework. We can mark events in different ways. For example, at the beginning of an edge or the end, but also right before the start of a steady state as seen in Figure 8. In this study, the events are marked right before they reach a steady state, which means right before the device is on or off. I believe this shows the complete event and allows a framework to detect the exact amount of energy consumed by an appliance during a certain event, therefore the Saxion Data is labeled in this manner.

Evaluation of the results
In this study, our results are generally a bit worse, except for the K-Means Clustering algorithm, which performed notably worse than in the referenced paper. A key difference could be the method used to determine the optimal number of clusters in the clustering algorithm. As described in the methodology, this study made use of the Silhouette method instead of the Elbow method to automate the process. This modification can significantly impact the results in case the outcome of the methods is different. Additionally, the dataset's characteristics and diversity of appliances included also play an important role, which is further explained in another discussion section. There is also the possibility that the author used different tuning parameters to suit the specific datasets or slightly adjusted the way the events are seen as correctly detected.

Apart from tuning the algorithm, there is also a preprocessing part before the event detection part. Although the same filter technique was used as in the proposed algorithm, there might be more effective preprocessing techniques for our specific use case. Understanding and realizing these nuances is important for interpreting the differences in performance.

Finally, the poor results of the K-Means Clustering algorithm could be explained according to the way the algorithm detects events. The K-Means Clustering algorithm initially groups all measurements into clusters, detecting events as the aggregate power signal transitions between these clusters. However, this method faces challenges with appliances like electric heaters, which show significant fluctuations in energy usage – often switching between full power and a near-off state for temperature regulation. Consequently, the aggregate power signal switches between clusters often, leading the algorithm to interpret these fluctuations as multiple small events, rather than a single major event. There are at least two ways to address this issue, either preprocess the signal to smooth out such fluctuations, though this could potentially lead to the loss of information in the signal. Alternatively, postprocessing could be used to combine closely occurring small events into a single major event. Both these options could lead to an improvement in precision and f1-score.

Automatic Hyperparameter Tuning
Determining the optimal number of clusters is part of a broader concept called Automatic Hyperparameter Tuning and is an expensive task with a time complexity of N^2, as can be seen in the results section. Consequently, the Silhouette method can take up to 30 minutes to determine the optimal number of clusters between 2 and 7 on a MacBook Pro 2018 for a single day of data.

Currently, there is still a lot of research done in this field to optimize the way of finding the optimal number of clusters. For example, the research done by Frank Huter to mimic the early termination of bad runs using a probabilistic model [14] or solving the problem with Sequential Model-based Bayesian Optimization (SMBO) [15].

The impact of the dataset on the results
The results of any study, especially in event detection studies, are heavily influenced by the dataset used. Each dataset has its unique characteristics such as noise levels, types of events and appliances, frequency of data collection, and overall data quality. These factors can have a significant impact on the performance of detection algorithms, which in turn affects the outcome of the study.

For instance, a dataset with high noise levels may lead to more false positives or negatives in event detection. Similarly, the frequency of data collection affects the level of detail captured by the event signatures, but it may also introduce more noise. The variety and complexity of events in the dataset also play a crucial role. Datasets with a wide range of event types, such as different appliance usages or varying durations, require more adaptable detection methods that may detect more false positives in other settings.

Therefore, the dataset used in the study must be representative of real-world scenarios for which the event detection system is designed. If the dataset does not accurately represent such scenarios, the results may not be applicable in such settings. This lack of representativeness could lead to overfitting, where the system performs well on a specific dataset but poorly in real-world applications.

Practical implications in a real-world setting
An effective event detection algorithm in energy monitoring has significant practical implications. Firstly, it can significantly improve energy management and efficiency. By accurately identifying when and how much energy is used by specific appliances, consumers and businesses can optimize their energy usage and align it with their energy production. This level of understanding allows for the implementation of smarter, more targeted energy-saving strategies, which can help reduce the stress on the grid.

Furthermore, precise event detection helps in maintaining and monitoring the health of electrical appliances. Detecting anomalies or changes in typical energy consumption patterns

can act as an early warning system for potential appliance malfunctions or inefficiencies, allowing for timely maintenance or replacement. In the context of smart homes and buildings, an effective event detection algorithm is essential to automate and optimize energy usage, contributing to the broader goal of creating more sustainable and energy-efficient living environments.

Further research
Further research following this event detection algorithms comparison could delve into several areas to eventually develop a comprehensive energy disaggregation framework. As can be seen in Figure 2 there are several steps to be taken before such a framework is finished. To make it easier for the event detection algorithms, one could delve deeper into preprocessing techniques. This includes noise filtering, data normalization, and possibly handling missing data, which can significantly improve the quality of input data and therefore improve the performance of the event detection algorithm.

Another research field would be the postprocessing after the events are detected. It is important to interpret all the events correctly to for example correctly estimate the total energy consumed per event or appliance. Furthermore, additional research on event detection algorithms could be done by delving deeper into the algorithms that are state-based or newly proposed as this Densely-connected Bi-directional Long Short-Term Memory method [16]. Finally, research into the integration of NILM systems with other smart home technologies and the broader smart grid infrastructure can lead to more comprehensive energy management solutions. This could include automated control systems that respond to event detection in real-time, optimizing energy use, and contributing to grid stability.

6 CONCLUSION

In conclusion, this comparative study on four event detection algorithms for energy disaggregation in a NILM context reveals several key findings. Noteworthy, the algorithms' performance characteristics are distinct, with the K-Means Clustering algorithm standing out due to its non-linear time complexity and its precision and F1-score falling significantly short compared to the other algorithms. On the other hand, the local threshold algorithm demonstrates a relatively strong precision and F1-score, whereas the adaptive threshold shows a good recall compared to the others. The implementations of the algorithms can be found on the GitHub page of the Ambient Intelligence (AMI) research group at Saxion University of Applied Sciences in Enschede. Since the Saxion Data is not a publicly available dataset, a request for access can also be sent to the Ambient Intelligence research group.

REFERENCES

[1] K. Brucke, S. Arens, J.-S. Telle, S. Schlüters, B. Hanke, K. von Maydell, and C. Agert, "Particle swarm optimization for energy disaggregation in industrial and commercial buildings," 06 2020.
[2] Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "Smart non-intrusive appliance identification using a novel local power histogramming descriptor with an improved k-nearest neighbors classifier," *Sustainable Cities and Society*, vol. 67, p. 102764, 02 2021.
[3] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," vol. 2, no. 150007, 2015.
[4] G. W. Hart, "Nonintrusive appliance load monitoring with finite-state appliance models. final report," 11 1995.
[5] A. Sankara, "Energy disaggregation in NIALM using hidden Markov models," 2014.
[6] O. S. Ajani, A. Kumar, R. Mallipeddi, S. Das, and P. N. Suganthan, "Benchmarking optimization-based energy disaggregation algorithms," *Energies*, vol. 15, p. 1600, Feb. 2022.
[7] J. Kelly and W. Knottenbelt, "Neural nilm: Deep neural networks applied to energy disaggregation," 11 2015.
[8] B. A. Tolnai, Z. Ma, and B. Jørgensen, "A scoping review of energy load disaggregation," pp. 209–221, 09 2023.
[9] N. ul Islam and S. Mehraj Shah, "A low complexity binary-weighted energy disaggregation framework for residential electricity consumption," *Energy and Buildings*, vol. 298, p. 113553, 2023.
[10] G. Pan, J. Qian, J. Ouyang, Y. Luo, and H. Wang, "Adaptive threshold event detection method based on standard deviation," *Measurement Science and Technology*, vol. 34, p. 075903, apr 2023.
[11] J. Alcalá, J. Ureña, Hernández, and D. Gualda, "Event-based energy disaggregation algorithm for activity monitoring from a single-point sensor," *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 10, pp. 2615–2626, 2017.
[12] H. Rafiq, P. Manandhar, E. Rodriguez-Ubinas, O. Ahmed Qureshi, and T. Palpanas, "A review of current methods and challenges of advanced deep learning-based non-intrusive load monitoring (nilm) in residential context," *Energy and Buildings*, vol. 305, p. 113890, 2024.
[13] J. Kolter and M. Johnson, "Redd: A public data set for energy disaggregation research," *Artif. Intell.*, vol. 25, 01 2011.
[14] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 3460–3468, AAAI Press, 2015.
[15] M. Feurer, J. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, Feb. 2015.
[16] N. Huang, H. Wang, X. Wang, C. Hu, and D. Wang, "A non-invasive load identification method considering feature dimensionality reduction and db-lstm," *Electronics*, vol. 13, no. 2, 2024.

A APPENDIX A

Algorithm 1: Event detection using k-means.

Input $\leftarrow P_{agg}(t)$
$P_{med} \leftarrow medianfilter(P_{agg}(t))$
$n_{clusters} \leftarrow SilhouetteMethod(P_{agg}(t))$
k-means $\leftarrow n_{clusters}, init = k - means++$
k-means $\leftarrow P_{med}$
return Clusterlabels
$C \leftarrow clusterlabels$
for *i in cluster labels* **do**
  if $C[i] \neq C[i-1]$ *and* $C[i] = C[i+1]$ **then**
    event index $\leftarrow i$;

B APPENDIX B

Algorithm 2: Local threshold based event detection.

Input $\leftarrow P_{agg}(t)$
$P_{med} \leftarrow medianfilter(P_{agg}(t))$
for *i in* P_{med} **do**
  $\Delta P \leftarrow p[i] - p[i-1]$
  $\Delta P \leftarrow abs(\Delta P)$
$n = 60$
for *i in* ΔP **do**
  $\Delta Power \leftarrow \Delta P[i : i+n]$
for *i in* $\Delta Power$ **do**
  $\mu[w] \leftarrow mean[j]$
  $\sigma[w] \leftarrow std[j]$
$s_p \leftarrow mean(\sigma[w]) + mean(\mu[w])$
$s_a = s_p/2$
$w_p \leftarrow np.where(\sigma[w] > s_a)$
for *i in* (w_p) **do**
  Perform peak detection
  $peaks \leftarrow loc[\Delta Power[i] > threshold]$
  where $Threshold \leftarrow a * \mu_w[i] + b * \sigma_w[i]$
  for *j in range(len(peaks[i]))* **do**
    $event\_index \leftarrow (n * w_p[i]) + peaks[i][0][j]$
Return event_index

C APPENDIX C

Algorithm 3: Adaptive threshold based event detection.

Input $\leftarrow P_{agg}(t)$
$P_{med} \leftarrow medianfilter(P_{agg}(t))$
for *i in* P_{med} **do**
  $\Delta P \leftarrow p[i] - p[i-1]$
  $\Delta P \leftarrow abs(\Delta P)$
$n = 60$
for *i in* ΔP **do**
  $\Delta Power \leftarrow \Delta P[i : i+n]$
for *i in* $\Delta Power$ **do**
  $\mu[w] \leftarrow mean[j]$
  $\sigma[w] \leftarrow std[j]$
$threshold \leftarrow \frac{24*\mu}{\sqrt{r}} + 0.4 * \sigma$
for *i, value in enumerate(threshold)* **do**
  Perform peak detection
  $peaks \leftarrow loc[\Delta Power[i] > value]$
  $selected\_peaks =$
   $peaks.where(\Delta P[i + peaks] > value)$
  for *j in selected_peaks* **do**
    $event\_index < i + w$
Return event_index

D APPENDIX D

Algorithm 4: Derivative-based event detection.

Input $\leftarrow P_{agg}(t)$
$P_{med} \leftarrow medianfilter(P_{agg}(t))$
$I_{rms} \leftarrow \sqrt{P_{med}/220V}$
for *i in* I_{rms} **do**
  $\Delta I_{rms} \leftarrow I_{rms}[i] - I_{rms}[i-1]$
$P_s \leftarrow \sqrt{\Delta I_{rms}}$
$P_s \leftarrow np.where(\Delta I_{rms} < 0, -P_s, P_s)$
r_edges
 $\leftarrow find\_peaks(P_s, distance = 1250, height = 0.45)$
f_edges
 $\leftarrow find\_peaks(-P_s, distance = 1250, height = 0.45)$
$edges \leftarrow np.concat(r\_edges, f\_edges)$
$event\_indices \leftarrow np.sort(edges)$
Return event_index

E APPENDIX E

Table 4. Testing Results - January 24, 2023

Metric	Local Threshold	Adaptive Threshold	Derivative Based	K-Means Clustering
FN	0.1181	0.0343	0.3157	0.1125
FP	0.2000	0.6488	0.2526	0.8600
TP	0.8000	0.3511	0.7473	0.1400
TN	0.8819	0.9657	0.6843	0.8875
Accuracy	0.9995	0.9979	0.9993	0.9954
Precision	0.8000	0.3511	0.7473	0.1400
Recall	0.8712	0.9108	0.7029	0.5544
F1-score	0.8341	0.5068	0.7244	0.2235

Table 5. Testing Results - January 25, 2023

Metric	Local Threshold	Adaptive Threshold	Derivative Based	K-Means Clustering
FN	0.2150	0.2105	0.4166	0.0353
FP	0.0215	0.0421	0.0952	0.8648
TP	0.9784	0.9578	0.9047	0.1351
TN	0.7850	0.7895	0.5834	0.9647
Accuracy	0.9997	0.9997	0.9995	0.9932
Precision	0.9784	0.9578	0.9047	0.1351
Recall	0.8198	0.8198	0.6846	0.7927
F1-score	0.8921	0.8834	0.7794	0.2309

Table 6. Testing Results - June 21, 2023

Metric	Local Threshold	Adaptive Threshold	Derivative Based	K-Means Clustering
FN	0.9997	0.1190	0.2876	0.1875
FP	0.9992	0.0119	0.0136	0.0250
TP	0.9995	0.9880	0.7206	0.9750
TN	0.9962	0.8810	0.7124	0.8125
Accuracy	0.9962	0.9998	0.9997	0.9998
Precision	0.9962	0.9880	0.9863	0.9750
Recall	0.9962	0.8924	0.7741	0.8387
F1-score	0.9962	0.9378	0.8674	0.9017

F APPENDIX F

Table 7. Evaluation Results - February 1, 2023

Metric	Local Threshold	Adaptive Threshold	Derivative Based	K-Means Clustering
FN	0.3235	0.0434	0.0962	0.1125
FP	0.4411	0.5217	0.6518	0.8600
TP	0.5588	0.4782	0.3481	0.1400
TN	0.6765	0.9566	0.9038	0.8875
Accuracy	0.9993	0.9992	0.9988	0.9954
Precision	0.5588	0.4782	0.3481	0.1400
Recall	0.6333	0.9166	0.7833	0.5544
F1-score	0.5937	0.6285	0.4820	0.2235

Table 8. Evaluation Results - February 2, 2023

Metric	Local Threshold	Adaptive Threshold	Derivative Based	K-Means Clustering
FN	0.2087	0.0283	0.3012	0.0353
FP	0.0769	0.0566	0.0602	0.8648
TP	0.9230	0.9433	0.9397	0.1351
TN	0.7913	0.9717	0.6988	0.9647
Accuracy	0.9996	0.9998	0.9996	0.9932
Precision	0.9230	0.9433	0.9397	0.1351
Recall	0.8155	0.9708	0.7572	0.7927
F1-score	0.8659	0.9569	0.8387	0.2309