Dynamic and Memory-Efficient Shape-Based Methodologies for User Type Identification in Smart Grid Applications

Rui Yuan, S. Ali Pourmousavi, *Senior Member, IEEE,* Wen L. Soong, *Fellow, IEEE,* and Jon A. R. Liisberg

***Abstract*—Behind-the-meter (BTM) equipment identification and monitoring their availability in real time in smart grid applications require computationally efficient methodologies suitable for edge deployment. Existing shape-based approaches, whilst maintaining interpretability advantages over structure-based methods, suffer from computational bottlenecks and memory constraints when processing streaming data. This paper develops three dynamic and memory-efficient updating strategies for Similarity Profile (SP) computation: an additive method for lossless updates, a fixed-memory approach with configurable inertia strategies, and a codebook-based technique employing dictionary learning for compressed data representation. The proposed methodologies eliminate the requirement for complete historical data reprocessing during each update, addressing critical limitations in edge computing environments. Comprehensive simulation studies using real-world photovoltaic (PV) user data demonstrate that the codebook-based approach achieves over 30% memory reduction while maintaining classification accuracy. The fixed-memory technique exhibits superior performance for applications requiring rapid change detection, with different inertia strategies providing varying sensitivity levels for diverse smart grid applications. These dynamic methodologies enable the practical deployment of interpretable BTM identification systems on resource-constrained edge devices whilst preserving the pattern recognition advantages of shape-based approaches.**

***Index Terms*—Dynamic updating, Data mining, Renewable energy, Pattern recognition, Binary classification, Time series mining, Data compression.**

NOMENCLATURE

$\hat{c_i}$	The number of similar days to day i.
$\hat{ch_i}$	Vector representing distance changes due to the updating process.
$\hat{d}_i$	Average dissimilarity of day i.
CR	Vector of the compressed representation of the whole time series data with a sequence of codewords in the codebook.
CW	Vector of codewords in the codebook, indexed by k.
SP′	Changes in the vector of Similarity Profile during updating, indexed by i.
SP	Vector of a Similarity Profile.
TS	Vector of metered electricity time series data by days, indexed by day i.
$\tilde{\text{SP}}$	Vector of a Similarity Profile of the old time series data before updating.
i	Index for daily data
k	Index for codewords
TH	A similarity threshold, i.e., day i and j are similar if $d_{i,j} \leq$ TH.
$c_{i,j}$	$c_{i,j} = 1$ if day i and j are similar. $c_{i,j} = 0$ otherwise.
$d_{i,j}$	Dissimilarity or distance between the time series of day i and day j.
d_{max}	The maximum dissimilarity or distance between all pairs of the energy time series.
d_{rep}	The dissimilarity or distance threshold to determine if two sub-patterns are interchangeable.
l	The total length of the energy time series data.
M	The number of days of data stored in the fixed memory method.
m	The number of time intervals for one day's worth of data.
N	The number of days in the whole energy time series data.
W	The number of the codewords in the Codebook-based method.

I. INTRODUCTION

In modern power systems, residential consumers equipped with rooftop PV systems, electric vehicles, and home batteries (known as prosumers) have changed the requirements for planning and operation. Traditional load patterns and volumes have changed drastically with the integration of behind-the-meter (BTM) generation and storage systems. In addition, prosumers who participate in demand response (DR) programs (such as virtual power plants) are dynamically contributing to local or wholesale markets and actively participating in the management of distribution grids. Consequently, it is imperative for various smart grid stakeholders, such as aggregators, retailers, distribution network operators, and market operators, among others, to have near real-time insights into the availability of these BTM resources.

For example, knowing how many prosumers under the jurisdiction of a retailer company currently have operational rooftop solar systems would improve aggregate demand estimation and allow for more accurate forecasts for grid operations. Similarly, distribution network operators require real-time insights into BTM resource availability to predict reverse power flows and implement appropriate voltage regulation strategies, and recently estimate dynamic import and export limits. Virtual power plant aggregators depend on accurate prosumer identification for optimal dispatch decisions and demand response program management. Although some countries maintain records of rooftop solar systems, documentation of other major BTM resources such as electric vehicles, battery

storage systems, and heat pumps is lacking [1]. Furthermore, the operational status of these resources could change due to equipment failures, maintenance, or behavioural modifications that require frequent updates. This underscores the necessity for developing BTM detection algorithms capable of operating effectively on low-resolution data using edge computing resources, for example existing smart meters, while preserving user privacy and minimising communication overhead.

Current studies on behind-the-meter (BTM) equipment identification methods can be categorised into two main aspects: their algorithmic approach and processing architecture [2], [3], [4]. From an algorithmic perspective, existing solutions fall further into structure-based and shape-based categories. Structure-based methods, including symbolic aggregation (SAX) and deep neural networks (DNNs), employ statistical transformations or black-box modelling that sacrifice interpretability for computational scalability. In contrast, shape-based approaches such as motif discovery analyse pattern similarity, preserving interpretability, which is a crucial feature for energy retailers and consumers, at the expense of higher computational complexity, particularly for extended time series data. Recent advances in pattern similarity computation have made shape-based methods increasingly viable for practical deployment.

Architecturally, these methods can be implemented using centralised or distributed processing paradigms [5]. Centralised approaches require full data aggregation at a central server, creating privacy concerns and storage burdens. Distributed implementations, such as federated learning schemes, address these issues by performing computations at edge nodes, but must remain structure-based and thus inherit their interpretability limitations. Furthermore, while distributed methods reduce the transmission of raw data, they still require substantial local computer power and communication bandwidth to exchange model parameters. The IRMAC framework introduced by the authors in [6] is a significant progress in distributed, shape-based solutions by extracting local pattern features and ensuring interpretability with a transparent classification method. However, all existing methods, including IRMAC require complete reprocessing of historical data during each update, which is a computationally prohibitive requirement for resource, constrained edge devices handling continuous data streams. This lack of incremental adaptability remains a critical bottleneck. This fundamental limitation motivates this work.

In this paper, we present a novel framework, shown in Fig.1, that combines the interpretability advantages of shape-based methods with a suitable dynamic computation for edge deployment. Within the proposed framework, three novel dynamic updating techniques are developed by the authors, that operate on streaming data without full historical reprocessing: an additive method for lossless updates, a fixed-memory method with configurable inertia strategies, and a codebook-based technique with dynamic distributed data storage. These solutions are fast, scalable, interpretable, and flexible. To validate the effectiveness of the proposed methods, we applied them to dynamically identify rooftop PV users and assessed their reaction to behavioural changes. Extensive simulation results across diverse prosumer scenarios demonstrate that these methods reduce memory and computation requirements by 30% to 50% compared to distributed computing solutions while maintaining equivalent accuracy. Thus, this paper establishes a new state-of-the-art for a practical, interpretable solution for BTM identification in edge computing environments. The novelty of this work includes the following.

- Development of three dynamic updating methods (additive, fixed-memory, codebook-based) to reduce memory and computation requirements.
- Comparison of different dropping strategies (low/medium/high-inertia) for the fixed-memory method, and compressing strategies for the codebook method, enabling trade-offs between short-term awareness and long-term stability, and privacy preservation for users altering their BTM applications.
- Using real-world PV data and synthetic benchmarks, demonstrating superior performance under dynamic prosumer behaviour (Section V).
- Thorough performance analysis on identification accuracy, memory savings, and sensitivity on users' behaviour changes for the proposed updating methods (Section IV).

This paper is organised as follows: Section II briefly explains the definitions of time series similarity measures, previous work on shape-based patterns mining, and the problem we are trying to solve in this paper. Section III details the three methods proposed in this paper for dynamic updating. The simulation studies are reported in Section IV, and the results are analysed in detail. Section V concludes the paper with a brief outline of prospective research directions.

II. Related Work

Identification of BTM equipment has become a critical focus in the smart grid era, where related research primarily addresses two challenges: distinguishing between users equipped with different BTM devices and tracking changes in device status or usage patterns over time. Shape-based solutions combine power system domain knowledge with data mining techniques, particularly pattern recognition and feature extraction, to uncover BTM usage features from long-term metered data [7], [8]. The following subsections summarise the key concepts and methodologies that underpin this work.

A. Time series similarity measures

Given an energy *time series* $E \in \mathbb{R}^l$, with length l, i.e., $e_l \in \mathbb{R} : E = (e_1, e_2, \cdots, e_l)$, the time series partition $E_n^m = (e_{n_1}, e_{n_2}, \cdots, e_{n_m})$ is a *sub-pattern* of E with a length of m. Given the context of residential energy usage, the length m in this paper refers to the length of a day, and n is the index of n^{th} day.

When searching for sub-patterns, we need *distance metric* to distinguish similar patterns from dissimilar ones. The dissimilarity between a pair of sub-patterns E_i^m and E_j^m is presented by $d_{i,j}$, which can be calculated by the L1 norm, Euclidean distance (ED) or Dynamic Time Warping (DTW), among other distance metrics [7], [8]. Considering the characteristics of residential electricity consumption, time shifting, and pattern

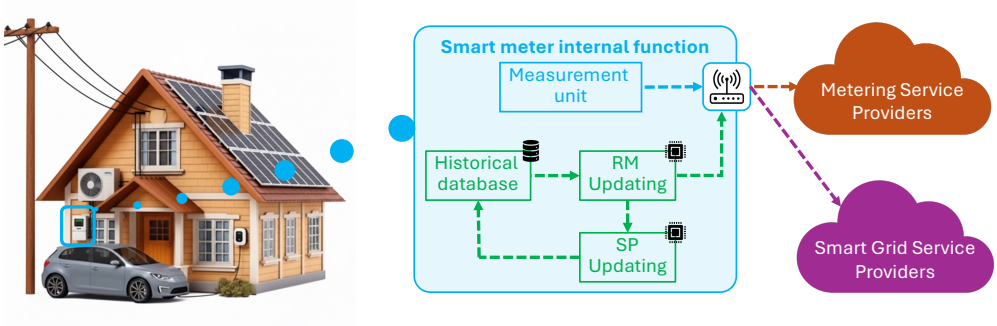


Fig. 1: High-level block diagram of the proposed methods showing a residential microgrid with smart meter and the existing and new functionalities

distortion due to irregular consumer behaviour, and to emphasise particular temporal patterns, we use annotated DTW as our distance metric [6]. This process of retrieving *sub-patterns* from long time series with *distance metric* is termed *similarity joins* [5], with applications in motif discovery.

B. Motif discovery and Matrix Profile

Motif is defined as the closest sub-pattern pair of the time series, minimising the *distance* [9]. The process of *motif discovery* applies *nearest-neighbour queries* to retrieve the closest neighbours of a given *sub-pattern* E_i^m, from the long energy time series E, i.e., $R_{1\text{NN}_q} = \min(d(E_i^m, E_j^m))$ for $\{E_i^m, E_j^m\} \in E$.

In 2016, a *nearest-neighbour query* based technique in [10], known as *MP*, was proposed that included the calculation of the z-normalised ED with the Fast Fourier Transform (FFT) to significantly decrease the spatial and temporal complexity of the motif discovery problem. *MP* retrieves the closest neighbours for each pair of sub-patterns with nearest-neighbour queries, and record the distance, i.e., $\mathbf{MP}[i] = R_{1\text{NN}_i}$, for $i \in [1, 2, 3, \cdots N]$.

C. Similarity profiles and refined motifs

In smart grid applications, it is commonly required to identify a single sub-pattern that is repeated the most frequently and exhibits the minimum distance from other sub-patterns, rather than simply identifying the nearest pair of sub-patterns. Therefore, the authors proposed a new definition of motif, called *Refined Motif (RM)*, in [6]. Compared to extracting the closest sub-patterns with *nearest-neighbour queries*, RM uses *range queries* to calculate how many sub-patterns are similar. Given a sub-pattern $E_n^m = (e_{n_1}, e_{n_2}, \cdots, e_{n_m})$, a full TS E, a distance measure d and a threshold ε, we can find the set of sub-patterns R in E that are within the distance ε from q, i.e., $R_{\text{RQ}_{E_n^m}} = \{E_i \in E | d(E_n^m, E_i) \leq \varepsilon\}$. This process is also called *Similarity self-join*. The RM can be subsequently determined using the *SP*. This involves retrieving range queries for each pair of sub-patterns and noting the number of matching patterns c_i for each query sub-pattern. In addition, it requires recording the maximum global distance $d_{\max}$ and the average distance for each range query $\hat{d}_i$. Consequently, SP forms a vector that represents the incidences of each sub-pattern throughout the entire series. It can be presented as in Fig. 2 and Equation (1), where $d_{i,j}$ is the annotated DTW distance between day i and day j, $c_{i,j}$ is a binary variable representing whether the two days are similar, the total similarity indices $\hat{c}_i$, which are integers representing the number of similar patterns with the current day i, and the average annotated DTW distance $\hat{d}_i$ normalised by $\max(d_{i,j})$ as the secondary impact factor for SP. It will differentiate the days with the same similarity indices.

	Day 1	Day 2	$\cdots$	Day N
Day 1		$< c_{12}, d_{12} >$	$\cdots$	$< c_{1N}, d_{1N} >$
Day 2	$< c_{21}, d_{21} >$		$\cdots$	$< c_{2N}, d_{2N} >$
$\vdots$	$\vdots$	$\vdots$		$\vdots$
Day N	$< c_{N1}, d_{N1} >$	$< c_{N2}, d_{N2} >$	$\cdots$	

$\Downarrow$

	Day 1	Day 2	$\cdots$	Day N
SP_i	SP_1	SP_2	$\cdots$	SP_N

Fig. 2: The proposed Similarity Profile (SP) in [6]

$$\text{SP}_i = \hat{c}_i - \frac{\hat{d}_i}{\max(d_{i,j})}. \tag{1}$$

Unlike MP assuming no domain knowledge of the data, SP takes into account domain knowledge to provide an interpretable and accurate solution for the identification of BTM appliances, as explained in [6]. As SP computation operates at the end of users, the authors in [6] introduced a distributed methodology named IRMAC to identify users with rooftop PV systems and those using electric heating systems. Simulation studies showed that IRMAC outperforms many intuitive and complex classification methods. For additional information, readers may consult [6].

D. Limitations in the existing approaches

Although IRMAC and other distributed approaches, such as MP and federated learning, offer clear benefits, they encounter a significant drawback: every update necessitates having access to the entire historical dataset and requires reprocessing. This

presents unresolved issues for the deployment of these methods at the edge, such as computational bottlenecks, memory overflows, and data privacy concerns.

In order to tackle these challenges, we suggest a range of solutions for streaming SP updates that remove the requirement for complete historical data and reprocessing, while integrating different levels of inertia-aware edge computing along with advantages in privacy-preserving compression. These methods are optimised to perform effectively under constrained computational power, storage, and communication bandwidth, and are validated using real-world smart meter data from rooftop PV households.

III. The Proposed Dynamic Update Methods

The block diagram in Fig. 1 presents the high-level architecture for the proposed dynamic update framework within typical Australian smart meter configurations. In these systems, half-hour consumption data is collected and transmitted daily to metering service providers, with retailers accessing the data for billing purposes.

The proposed framework could operate within existing smart meters or dedicated edge devices, using the availability of the measurement unit's daily data to trigger SP updates. Rather than reprocessing complete historical datasets, as required by existing methods, our approach updates the end-user's SP through single-iteration calculations using new daily data. This significantly reduces memory and computational demands, decreases the risk of privacy breaching of full historical data, while enabling real-time RM extraction and communication to smart grid service providers, as shown in Fig. 1.

This paper develops three novel dynamic updating methodologies within the proposed framework specifically designed for edge-based SP computation: the additive method provides lossless updates with increasing memory requirements; the fixed-memory method maintains constant memory allocation through configurable data retention strategies; and the codebook-based method employs dictionary learning for compressed data representation with dynamic memory allocation. These methods address different operational requirements and resource constraints, making them suitable for various smart grid applications.

The additive method ensures complete accuracy by preserving all historical information, although memory consumption increases linearly with time. The fixed-memory method offers three inertia strategies—low, medium, and high—that provide varying levels of sensitivity to detect behavioural changes while maintaining fixed computational complexity and memory requirement. The codebook-based method achieves optimal long-term performance by balancing accuracy with memory efficiency through adaptive compression techniques, as demonstrated in simulation studies.

The detailed implementation, performance characteristics, and application scenarios of each method are presented in the following subsections.

A. RM update with Additive method

Using January as an example, the additive method provides a baseline approach for dynamic SP, represented schematically in Fig. 3. The daily profile for each day requires a fixed memory allocation of $1X$ bytes, where X is the memory requirement for the demand profile for one day. The required memory grows daily.

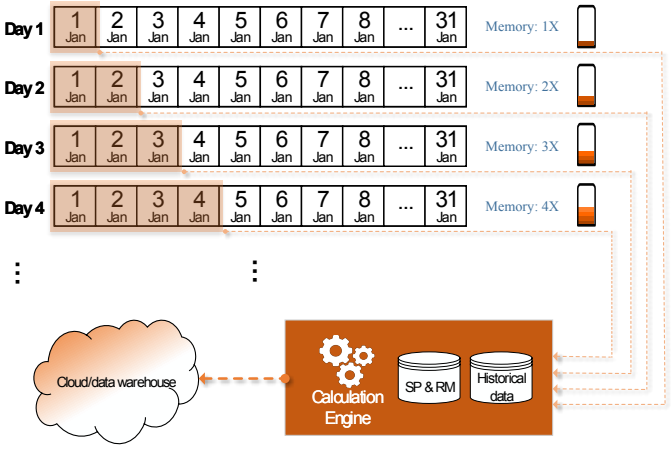


Fig. 3: A block diagram showing the operation of the additive method and its memory requirements

The incoming data is compared with all existing historical patterns to update $\overrightarrow{\text{SP}}$ with changes $\overrightarrow{\text{SP}'}$. Therefore, the updated time series contains all the daily data from day 1 to the day before. After updating, the new time series and SP will be stored at the user's end, where the RM can be updated from the new SP, as shown in Fig. 2. It can be seen in the figure that the memory requirement will grow constantly over time. Equations (2-5) present the additive updating process from day N to day $N+1$ in three steps. In the first step, the global maximum dissimilarity d_{max} is calculated with the distance between day $N+1$ and days 1 to N. Second, changes in SP, that is, SP'_i, are calculated based on the global maximum dissimilarity, and the counters are updated by the dissimilarity. Third, SP is calculated by updating the old SP value, $\tilde{\text{SP}}_i$, with the new incoming value, SP_{N+1} as in Equations (4-5). The procedure takes one iteration to update the RM with linear complexity.

$$\max(d_{i,j}) = \max(\tilde{d}_{max}, d_{i,N+1}), \quad 1 \le i \le N, \tag{2}$$

$$\text{SP}'_i = c_{i,N+1} - \frac{d_{i,N+1} \cdot \tilde{d}_{max}}{\max(d_{i,j})} + \frac{(\lceil \tilde{\text{SP}}_i \rceil - \tilde{\text{SP}}_i) \cdot (N-1)}{N}, \quad 1 \le i \le N, \tag{3}$$

$$\text{SP}_{[1:N]} = \text{SP}'_{[1:N]} + \tilde{\text{SP}}_{[1:N]}, \tag{4}$$

$$\text{SP}_{N+1} = \sum_{i=1}^{N} c_{i,N+1} - \frac{\sum_{i=1}^{N} d_{i,N+1}}{\max(d_{i,j}) \cdot N}. \tag{5}$$

Algorithm 1 shows a step-by-step computation process. For a time series with length l and window size m, which grows daily, recomputing the SP takes $O(l^2 m)$ time complexity, as discussed in [6]. On the other hand, updating an incoming

day's data at the user's end with the additive method on the same data length takes the time complexity of $O(lm)$.

Algorithm 1 Additive Method for Updating SP

Require: TS contains N days data $\text{TS}_{[1:N]}$ with a new incoming data TS_{N+1} to be updated, SP with N days' record $\text{SP}_{[1:N]}$, a maximum dis-similarity d_{max}, and a threshold TH

$c_{[1:N]} \leftarrow [0, 0, \dots 0]$ ▷ initialise counters
$ch_{[1:N]} \leftarrow [0, 0, \dots 0]$ ▷ initialise changes of distances
$\hat{d}_{N+1} \leftarrow 0$ ▷ initialise dissimilarity for new data
for $i = 1, 2 \dots N$ **do**
  $d_{i,N+1} \leftarrow DTW(\text{TS}_{N+1}, \text{TS}_i)$
  $\hat{d}_{N+1} \leftarrow \hat{d}_{N+1} + d_{i,N+1}$
  $d'_{max} \leftarrow \max(d_{max}, d_{i,N+1})$ ▷ update max distance
  $ch_i \leftarrow d_{i,N+1}$
  if $d_{i,N+1} \leq \text{TH}$ **then** ▷ check threshold
    $c_i \leftarrow c_i + 1$
    $c_{N+1} \leftarrow c_{N+1} + 1$
  end if
end for
$avg_{scaled} \leftarrow \lceil \text{SP}_{[1:N]} \rceil - \text{SP}_{[1:N]}$
$avg' \leftarrow \frac{avg_{scaled} \cdot (N-1) \cdot d_{max} + ch_{[1:N]}}{d'_{max} \cdot N} - avg_{scaled}$
$\text{SP}_{[1:N]} \leftarrow \text{SP}_{[1:N]} + c_{[1:N]} - avg'$ ▷ Equation (3-4)
$\text{SP}_{N+1} \leftarrow c_{N+1} - \frac{d_{new} \cdot (N+1)}{d'_{max} \cdot N}$ ▷ Equation (5)
return SP, TS, d'_{max} ▷ RM is TS_i where $\text{SP}_i == \max(\text{SP})$

The main bottleneck of the additive method is that memory and computational time requirements increase with the number of days, as shown in Fig. 3. In addition, the additive method provides a global RM because of using the entire historical data. Therefore, it can be as accurate as the method proposed in [6]. However, this method is less sensitive to identifying the user's type switching, i.e., a non-PV user switching to a PV user or vice versa. To address these issues, we propose a modified updating approach in the next subsection.

B. RM update with Fixed-memory method

The fixed-memory method addresses the unbounded memory growth of the additive approach by maintaining a sliding window of the new incoming sub-pattern as shown in Fig. 4. New daily data replaces data from a selected past day in the memory, thereby maintaining a consistent memory allocation size, denoted M. Compared to the additive method, the time complexity of the fixed-memory method is $O(M \cdot m)$, which is l/M times the complexity of the Additive method since l is normally much larger than M. Algorithm 2 illustrates the sequential procedure of the fixed-memory method.

Algorithm 2 Fixed-memory method

Require: variable TS contains M stored daily sub-patterns $\text{TS}_{[1:M]}$ with an incoming day's data TS_{M+1} to update, variable SP with M days' similarity information $\text{SP}_{[1:M]}$, a maximum dis-similarity d_{max}, and a threshold TH

$\text{TS}_{in} \leftarrow \text{TS}_{M+1}$ ▷ new incoming data

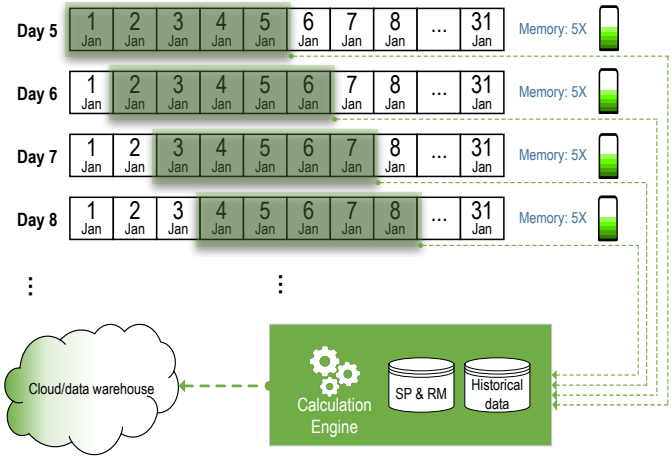


Fig. 4: A block diagram representing the operation of the fixed-memory method and its memory requirement

$\text{TS}_{out} \leftarrow \text{TS}_{index}$ ▷ drop data by index
Remove TS_{index} from TS
$c_{[1:M]} \leftarrow [0, 0, \dots 0]$ ▷ initialise counters
$ch_{[1:M]} \leftarrow [0, 0, \dots 0]$ ▷ initialise changes of distances
$d_{new} \leftarrow 0$ ▷ initialise dissimilarity for new data
for $i = 1, 2, 3 \dots$ M-1 **do**
  $d_{out} \leftarrow \text{DTW}(\text{TS}_{out}, \text{TS}_{[i]})$
  $d_{in} \leftarrow \text{DTW}(\text{TS}_{in}, \text{TS}_{[i]})$
  $d_{new} \leftarrow d_{new} + d_{in}$
  $d'_{max} \leftarrow \max(d_{max}, d_{in})$ ▷ update distance
  $ch_i \leftarrow d_{in} - d_{out}$
  $ch_i \leftarrow \frac{ch_i}{M}$ ▷ changes in average
  if $d_{out} \leq TH$ **then** ▷ check threshold
    $c_{[i]} \leftarrow c_{[i]} - 1$
  end if
  if $d_{in} \leq TH$ **then** ▷ check threshold
    $c_i \leftarrow c_i + 1$
    $c_M \leftarrow c_{M+1} + 1$
  end if
end for
$avg_{scaled} \leftarrow \lceil \text{SP}_{[1:M-1]} \rceil - \text{SP}_{[1:M-1]}$
$avg' \leftarrow \frac{avg_{scaled} \cdot d_{max} + ch_{[1:M-1]}}{d'_{max}} - avg_{scaled}$
$\text{SP}_{[1:M-1]} \leftarrow \text{SP}_{[1:M-1]} + c_{[1:M-1]} - avg'$
$\text{SP}_M \leftarrow c_M - \frac{d_{new}}{d'_{max}}$
return SP, TS, d'_{max} ▷ RM is TS_i where $\text{SP}_i = \max(\text{SP})$

This method is a lossy solution in terms of preserving the actual SP values due to the method tracking the maximum distance within the memory as opposed to the global maximum distance, i.e., $\max(d_{i,j})$ in Equation (2). The extracted RM is expected to be accurate because the decimal relationship in the SP is preserved, with its decimal part scaled at the same level in Equation (5). There are two hyperparameters in the fixed-memory method, i.e., sub-pattern dropping strategy and the window size, that are discussed next.

1) Different dropping strategies: The performance of the fixed-memory method is critically dependent on the strategy to select TS_{out}. Three dropping strategies are evaluated, which

could be used in different smart grid applications as follows:

1) **Low-inertia strategy**: In this approach, we remove the oldest day profile from memory to save the incoming daily data. In this way, we can access the latest M days of consumer demand profiles. As a result, this strategy is heavily influenced by the most recent historical data and the temporal dynamics of it will play the main role in finding RM. In other words, it offers rapid detection of behaviour changes, but potentially discards valuable historical patterns.
2) **High-inertia strategy**: As daily time series and SP are maintained and do not have to be sequential, we suggest implementing a "high-inertia strategy" that eliminates data with minimal SP value to maintain long-term memory. This strategy will preserve similar patterns in the memory. Therefore, it preserves dominant motifs at the cost of reduced sensitivity to subtle changes.
3) **Medium-inertia strategy**: Taking into account the issues of the previous reduction strategies, we propose the third strategy in which we eliminate daily data with the median SP value. The idea is to keep the most similar and dissimilar patterns in the database. In this way, the medium-inertia strategy maintains diversity in the stored patterns while filtering outliers.

To better explain the three strategies, consider an example in which a consumer's PV solar system malfunctions, resulting in a temporary shift from a solar user to a non-solar user, as shown in Fig. 5. The yellow and light-blue areas in the figure represent the periods before and after the breakdown of the solar system. It can be seen that it takes three and four updates for the low-inertia and medium-inertia strategies, respectively, to identify the breakdown (shown by the vertical line). However, the high-inertia strategy cannot detect this change even after four updates. Therefore, it can be concluded that the low-inertia strategy method is good at tracking even the smallest changes in user behaviour. In contrast, the high-inertia strategy works best for users with stable behaviour. A comprehensive simulation study on the capabilities of the three methods is presented in Section IV.

2) Hyperparameter justification for memory size: The performance of the fixed-memory method depends critically on the parameter M, which determines how many daily sub-patterns are preserved in memory. Although larger M retains more historical patterns at increased memory and computation costs, smaller values improve efficiency but risk losing important temporal features. This trade-off is particularly relevant for edge devices with limited computational resources and storage. To guide practitioners in selecting M, Section IV provides a systematic analysis that examines: (1) accuracy vs. memory trade-off for different values of M; (2) identification of the knee point where accuracy gains diminish; and (3) practical recommendations based on application requirements.

C. RM update with Codebook-based method

The codebook-based method provides an adaptive solution for dynamic RM updates by employing dictionary learning techniques to achieve memory efficiency while preserving pattern fidelity [11], [5]. This method is especially appropriate for long-duration applications where memory limitations are significant, but it is necessary to preserve full historical data in a compressed format. The method maintains two key data structures, namely Codeword and representations. The codeword is a sub-pattern cw_i in the time series representing a set of other similar sub-pattern neighbours whose distance is smaller than a replaceable distance threshold d_{rep}, that is, $\{E_i \in E | d(\mathrm{cw}_i, E_i) \leq d_{\text{rep}}\}$. Representation cr_i is the pointer that maps the compressed time series partition i to the codeword cw_s of the codebook, that is, $E_i{}' = D \cdot \mathrm{cr}_i = \mathrm{cw}_s$. We adopt the same idea for the real-time updating problem of RM discovery, as shown in the block diagram of Fig. 6. Since the dissimilarity of the daily patterns in our proposed dynamic RM is measured independently, the RM discovery process does not require the sub-patterns to be in a temporal sequence. Consequently, we propose two sub-strategies for the codebook method, which are discussed in the following subsections.

1) Codebook method with compressed representation: The first strategy is the conventional codebook method. In this method, the codebook and the historical SP are stored at the user's end. The distance between the new incoming data and the existing codewords in the codebook is computed. If the distance of the new incoming data and the most similar codeword is smaller than a pre-defined threshold, it will be replaced with an existing codeword. Otherwise, it will be added to the codebook as a new codeword. In general, this method will compress the end-user's data with a codebook and recover the data with compressed representation before the updating process. The pseudocode for this approach is presented in Algorithm 3.

Algorithm 3 Codebook with compressed representation

Require: variable CW contains W stored codewords $\mathrm{CW}_{[1:W]}$ with a incoming TS_{in} to update, variable SP with N days' similarity $\mathrm{SP}_{[1:N]}$, a compressed representation with N day's code number $\mathrm{CR}_{[1:N+1]}$, a maximum dissimilarity d_{max}, and a distance value which is seen as replaceable d_{rep}

```
c_[1:N] ← [0, 0, ... 0]                          ▷ initialise counters
ch_[1:N] ← [0, 0, ... 0]      ▷ initialise changes of distances
d_new ← 0          ▷ initialise dissimilarity for the new data
d_min ← d_rep
TS_[1:N] ← CW_CR_[1:N]                            ▷ recover TS
CR_{N+1} ← W + 1                                ▷ initialise codebook
for i = 1, 2, 3 ... N do
    d_in ← DTW(TS_in, TS_i)
    d_new ← d_new + d_in
    d'_max ← max(d_max, d_in)                          ▷ update d
    ch_[i] ← d_in
    if d_in ≤ TH then                              ▷ check threshold
        c_i ← c_i + 1
        c_{N+1} ← c_{N+1} + 1
    end if
    if d_in ≤ d_min then                        ▷ check if replaceable
        d_min ← d_in
        CR_{N+1} ← CR_i
```

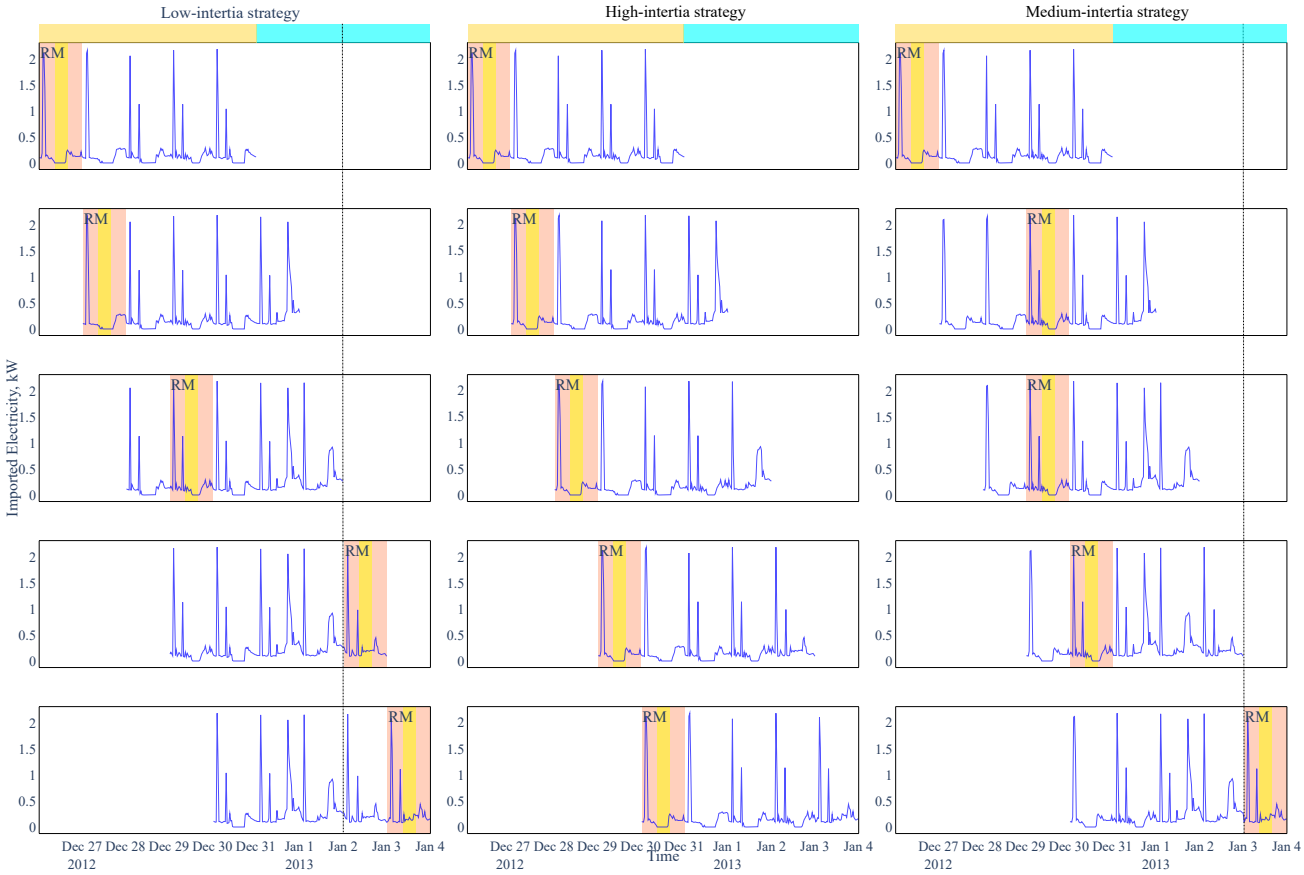


Fig. 5: Comparison of three variations of the fixed-memory method in detecting solar type switching, window size $= 5$

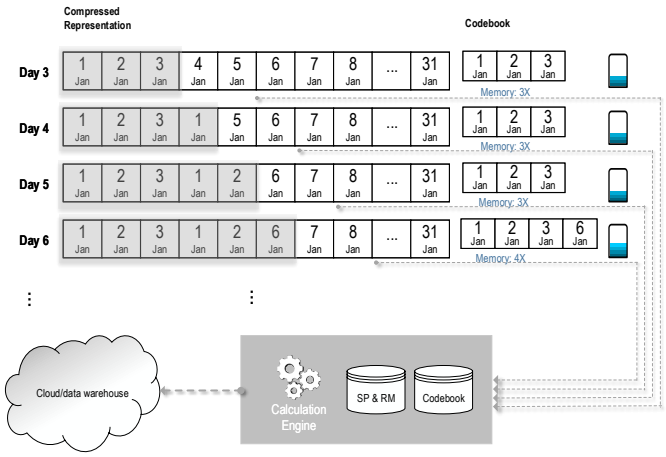


Fig. 6: A bock diagram illustrating the Codebook method for dynamic RM discovery

end if
end for
$avg_{scaled} \leftarrow \lceil \text{SP}_{[1:N]} \rceil - \text{SP}_{[1:N]}$
$avg' \leftarrow \frac{avg_{scaled} \cdot (N-1) \cdot d_{max} + ch_{[1:N]}}{d'_{max} \cdot N} - avg_{scaled}$
$\text{SP}_{[1:N]} \leftarrow \text{SP}_{[1:N]} + c_{[1:N]} - avg'$
$\text{SP}_{N+1} \leftarrow c_{N+1} - \frac{d_{new}}{d'_{max}}$
if $\text{CB}_{N+1} == W + 1$ **then**
$\text{CR}_{W+1} \leftarrow \text{TS}_{in}$
end if
return SP, CW, d'_{max}, CR ▷ RM is CW_{CR_i} where $\text{SP}_i == \max(\text{SP})$

2) Codebook without compressed representation: In contrast to the earlier codebook approach, we introduce a novel method that presumes reconstructing the original time series from codewords is not required. Instead, an alternative codebook known as the Patterns Dictionary (PD) is stored at the end-user's location. PD is proposed to store the codewords as keys and the occurrences as values. The end user can maintain and revise the complete SP based on codewords and occurrences, though it lacks a temporal aspect since the SP's index values do not indicate collection times. Compared to the Codebook model with compressed representation, this strategy preserves the relationship of each sub-pattern but in an anonymous way because the temporal sequences are not included. More specifically, the SP's sub-pattern with a higher index does not necessarily refer to newer data than the sub-pattern with a lower index. Furthermore, it requires less memory since we do not need a compressed representation. The implementation of this strategy is explained step by step in Algorithm 4.

Algorithm 4 Codebook method without compressed representation

Require: a dictionary with W stored daily patterns as keys and occurrences as values, i.e. PD $= \{\text{CR}_1 : n_1, ...\text{CR}_W : n_W\}$, an incoming TS_{in} to update, variable SP with N days $\text{SP}_{[1:N]}$, a maximum dis-similarity d_{max}, and a value of distance which can be seen as pre-defined d_{rep}

$c_{[1:N+1]} \leftarrow [0, 0, \dots 0]$ ▷ initialise counters
$ch_{[1:N]} \leftarrow [0, 0, \dots 0]$ ▷ initialise changes of distances
$d_{new} \leftarrow 0$ ▷ initialise dissimilarity for new data
$d_{min} \leftarrow d_{rep}$
$p \leftarrow W + 1$ ▷ initialise the position for new data in PD
$cr \leftarrow 1$ ▷ track the position in SP
for $i = 1, 2, 3 \dots W$ **do**
  $d_{in} \leftarrow DTW(\text{TS}_{in}, \text{CR}_i)$
  $n_i \leftarrow \text{PD}_{\text{CR}_i}$ ▷ occurrence of the data
  $d_{new} \leftarrow d_{new} + d_{in} \cdot n_i$
  $d'_{max} \leftarrow \max(d_{max}, d_{in})$ ▷ update d
  $ch_{[cr:cr+n_i]} \leftarrow d_{in}$
  if $d_{in} \leq TH$ **then** ▷ check threshold
    $c_{[cr:cr+n_i]} \leftarrow c_{[cr:cr+n_i]} + 1$
    $c_{N+1} \leftarrow c_{N+1} + 1$
  end if
  if $d_{in} \leq d_{min}$ **then** ▷ check if replaceable
    $d_{min} \leftarrow d_{in}$
    $p \leftarrow i$
  end if
  $cr \leftarrow cr + n_i$ ▷ update the position
end for
$avg_{scaled} \leftarrow \lceil \text{SP}_{[1:N]} \rceil - \text{SP}_{[1:N]}$
$avg' \leftarrow \frac{avg_{scaled} \cdot (N-1) \cdot d_{max} + ch_{[1:N]}}{d'_{max} \cdot N} - avg_{scaled}$
$\text{SP}_{[1:N]} \leftarrow \text{SP}_{[1:N]} + c_{[1:N]} - avg'$
$\text{SP}_{in} \leftarrow c_{[N+1]} - \frac{d_{new}}{d'_{max}}$ ▷ new data's SP
if $p \leq W + 1$ **then**
  $n_p \leftarrow n_p + 1$
  $cr \leftarrow n_1 + n_2 + \dots n_p$ ▷ find the position to insert new SP
  $\text{SP}_{[cr+1:N+1]} \leftarrow \text{SP}_{[cr:N]}$ ▷ shift one place for new data
  $\text{SP}_{[cr]} \leftarrow \text{SP}_{in}$ ▷ insert new SP
else
  $\text{CR}_{W+1} \leftarrow \text{TS}_{in}$
  $n_{W+1} \leftarrow 1$
  $\text{SP}_{N+1} \leftarrow \text{SP}_{in}$
end if
return SP, PD $= \{\text{CR}_1 : n_1, ...\}$, d'_{max} ▷ RM is CR_i where $\text{SP}_{[n_1+..n_i]} == \max(\text{SP})$

IV. Simulation Studies

To assess the performance of our proposed methods, the Solar Home data from New South Wales, Australia, is used for the simulation studies, which contains half-hourly PV generation and demand data from 300 residential consumers, spanning three continuous years from 2010–11 to 2012–13 [12]. Our evaluation focuses on three key aspects: (1) PV user identification accuracy, (2) computational efficiency, and (3) memory requirements. Since both the fixed-memory and codebook-based methods have different sub-strategies, see Sections III-B and III-C, we report and discuss the results of the fixed-memory and codebook models individually to better show their advantages and limitations. Finally, the overall accuracy of all the proposed methods is compared in Section IV-C for different data lengths.

A. Fixed-memory method: evaluation on dropping strategies and memory sizes

In order to thoroughly assess the fixed-memory approaches, a comprehensive simulation study was carried out where users transitioned between being solar and non-solar users. We selected 100 random users with rooftop solar systems from the dataset, assuming their systems would fail on day 10, making them non-solar users afterward. Also, another dataset with 100 non-solar users was created, where they acquired solar systems on day 10, resulting in their conversion to solar users for the remaining duration of the study. The frequency of daily RM updates was tracked to observe the performance of different methods to detect user transitions. Fig. 7 and Fig. 8 illustrate the results for solar users becoming non-solar users and vice versa.

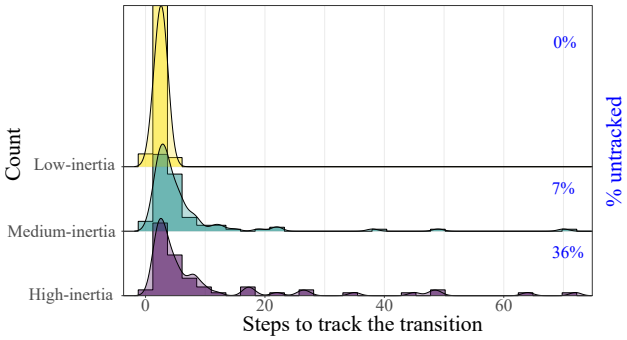


Fig. 7: Inertia study for 100 non-solar users becoming solar users: median number of iterations to detect the change is 3, 4 and 11 for low-, medium-, and high-inertia strategies

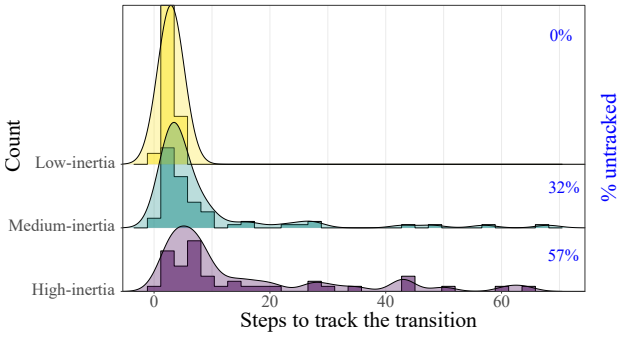


Fig. 8: Inertia study for 100 solar users becoming non-solar users: median number of iterations to detect the change is 3, 7 and 80 for low-, medium-, and high-inertia strategies

The simulations reveal distinct performance characteristics across the three inertia strategies in tracking user-type transitions. The low-inertia strategy shows remarkable responsiveness, achieving 100% detection of transitions between solar and non-solar states. Conversely, the high-inertia strategy effectively retains patterns but poorly detects changes, missing 36% of non-solar-to-solar and 57% of solar-to-non-solar transitions. The medium-inertia strategy strikes a balance, detecting over 50% of transitions within 30 days while maintaining reasonable long-term stability. The differences in performance are attributed to core design variations: the sliding window method of low-inertia favours recent information, the feature-preserving strategy of high-inertia emphasises on detecting stable patterns, and the selective retention in medium-inertia ensures the maintenance of diversity. Notably, all strategies perform better at detecting non-solar-to-solar transitions (7-32% unidentified) versus solar-to-non-solar (36-57% unidentified), as solar patterns exhibit more distinctive SP signatures that are easier to preserve in memory-limited scenarios.

We also ran another simulation study with different memory sizes to assess its impact on the performance of the fixed-memory approaches, i.e., identifying solar users from non-solar users. The results are shown in Fig. 9. The low-inertia strategy shows strong memory dependence, with accuracy improving from 82% to 91% as memory expands from 5 to 30 days. The high-inertia strategy achieves the highest average accuracy (94%), demonstrating stable long-term feature extraction. Medium-inertia achieves peak accuracy with 10 days of memory size, validating its balanced design on tracking transitions and preserving featured patterns.

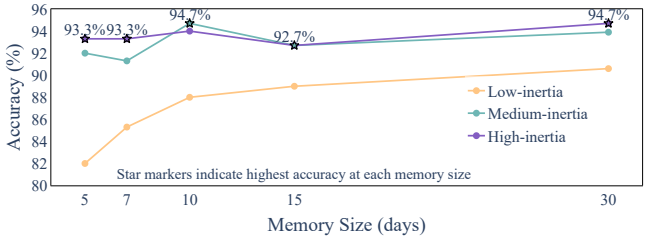


Fig. 9: Performance of the three dropping strategies

B. Codebook: Compression rate and accuracy

The compression performance is quantified by *memory saving*, which is the ratio between saved memory versus total memory. A sensitivity analysis is conducted for 300 consumers with 1, 2 and 3 months of data from summer. The results are summarised in Table I. It can be seen that the memory saving rate increases when more data is included. In other words, adding more data may lead to more stable codewords.

One hyperparameter in the codebook-based method is the threshold to determine if one sub-pattern can be replaced with the codeword. A lower threshold results in more codewords in the codebook or PD; hence, less memory saving. On the other hand, the distance between replaced sub-patterns and the codewords will be smaller. As a result, there is a trade-off between memory saving and accuracy. In particular, the compression performance for each user is different because of the variability in their patterns. To measure the trade-off between accuracy and compression, we performed a simulation study in which the threshold is changed from 0.5 to 2 to compute the memory saving for the 300 users. In addition, we measured the performance of the extracted RMs by identifying solar users with a linear classifier developed in our previous work [6]. Fig. 10 illustrates the memory saving distributions for end-users with varying classification accuracy.

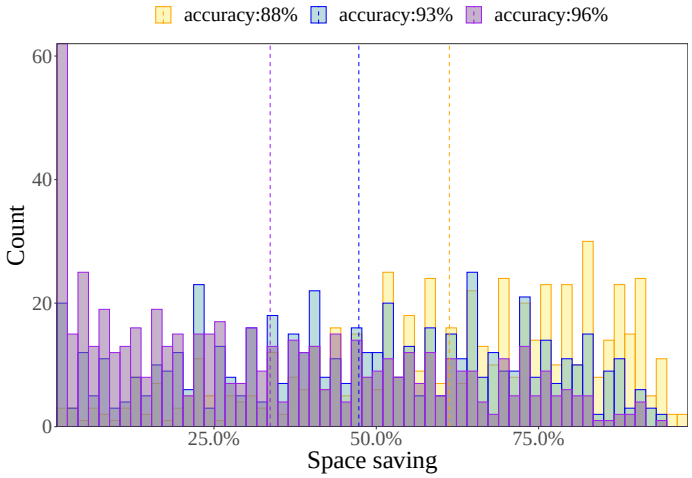


Fig. 10: Trade-off between memory saving and accuracy in the codebook-based method for 299 solar users and 299 non-solar users

In general, as illustrated in Fig. 10, memory saving exhibits considerable variability between users. At 96% accuracy, 87 users barely show any memory saving. However, most user data can be compressed to a certain degree, with an average memory reduction of 30%. As accuracy declines, the distribution shifts toward greater memory savings. At 88% accuracy on the classification problem, most users achieve more than 50% memory saving.

Fig. 11 presents the Pareto front derived from additional experiments on the trade-off between average memory savings and accuracy, using data collected over a 3-month period for 598 users. The fixed-memory method with the three elimination strategies is also included for performance comparison.

From Fig. 11, it is apparent that the codebook method without using compressed representation managed to achieve an average of 30% memory savings without compromising accuracy. At compression levels exceeding 50% memory savings, the fixed-memory method's high- and medium-inertia strategies perform substantially better. In contrast, as indicated

TABLE I: Compression ratio of the codebook method

Month \ Memory saving	min	max	average
1	9%	96%	30%
2	8%	98%	31%
3	6%	98%	34%

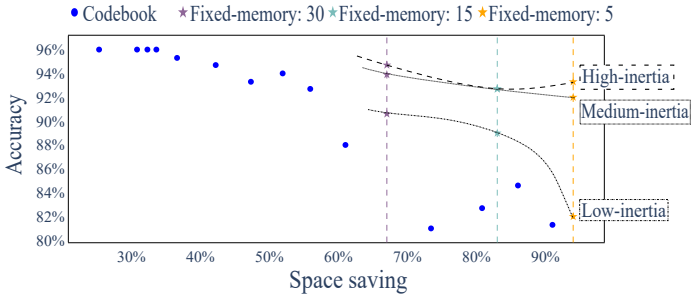


Fig. 11: Accuracy vs. memory saving Pareto front for codebook and fixed-memory techniques

in Table I, the memory savings in the codebook method increase with longer time series. This suggests that accuracy in the high compression range may improve with extended historical data availability.

C. Accuracy analysis for the proposed updating methods

The overall accuracy and memory size are presented in Table II, where the memory size in the fixed-memory method is 15 (i.e., $M = 15$) and the compression rate of the codebook method is 30%.

TABLE II: Performance of the different methods in the solar user identification problem

Month \ Accuracy	Additive	Fixed-Memory (low)	Fixed-Memory (high)	Fixed-Memory (medium)	**Codebook**
1	**94.0%**	88.0%	92.7%	92.0%	**94.0%**
2	94.7%	86.0%	94.7%	**96.0%**	**96.0%**
3	**96.0%**	89.3%	92.7%	92.7%	**96.0%**

The codebook and additive methods show the overall best accuracy among the three methods, followed by the fixed-memory with high-inertia strategy. The accuracy of the fixed-memory method with a low-inertia strategy significantly varies with different data. Based on simulation studies in previous sections, the main features of the three updating methods are summarised in Table III.

D. Summary of the Performance Analysis and Implications

The simulation results reveal different performance characteristics between methodologies. The 96% accuracy of the additive method comes at the cost of linear memory growth, making it unsuitable for long-term deployment. The fixed-memory approach shows remarkable efficiency gains, reducing computational complexity from $O(lm)$ to $O(Mm)$, where $M \ll l$. This represents an 85-90% reduction in processing requirements for typical deployment scenarios.

The performance of the codebook-based method exhibits data-dependent characteristics. For users with highly repetitive patterns, compression ratios exceed 60%, while diverse consumption profiles achieve 20-30% compression, without compromising the accuracy. This variability suggests the suitability of the method for different consumer segments, with industrial or commercial applications likely benefiting from higher compression rates due to more predictable usage patterns.

V. Conclusion

This paper proposes three RM updating methods and six sub-strategies for smart grid applications. The performance and sensitivity of each method to its respective hyperparameters are analysed with a set of simulation studies. In general, the additive method yields the most accurate outcomes, although with an increased demand for memory and computational resources. The fixed-memory method offers the highest memory saving and fixed temporal complexity. The three data elimination strategies are proposed that can be used in different circumstances. The low-inertia strategy is sensitive to behavioural changes and memory size; hence, it is good for tracking changes. The high-inertia strategy cannot detect sudden changes but is good at detecting similar sub-patterns in daily data. The medium-inertia strategy is a balanced option in terms of accuracy, memory saving, and detecting changes. The codebook method is the most balanced solution, providing more than 30% of memory saving without sacrificing accuracy. With this method, both the codebook with compressed representation and PD provide the same RM, but the former can easily recover the whole time series while the latter is anonymous. The memory savings of the codebook method increase with the data size, making it a more robust solution for storing long-term patterns or high-resolution data for end users. Another significant advantage of the codebook method is the privacy-preserving feature at the user's end.

The authors envision three possible directions for future work. Firstly, the memory size for the fixed-memory method and the pre-defined threshold for the codebook-based method were chosen based on educated guesses. These hyperparameters require systematic optimisation as they are fundamentally tied to data resolution and application requirements. We plan to develop generalised selection strategies through extended sensitivity analysis across diverse datasets. Secondly, the extracted RM can be used in more applications, e.g., demand modelling, forecasting, and demand-side management measures. For instance, it can help detect new EV users, new Air Conditioners or heat pumps, or shifts between cooling and heating regimes in near real-time. It could help detect malicious behaviours, such as electricity theft. Future research could focus on broadening the application of the proposed framework to additional smart grid challenges. Third, our study is limited to the data from the utility meters that are accessible to aggregators and retailers. With the availability of high-resolution data from smart meters, combined with our memory-efficient updating strategies such as the fixed-memory or codebook method, it becomes possible to identify a larger number of appliances, providing better insight into consumer behaviour and variations without memory outage concern [13], [14].

References

[1] G. Barbose, N. Darghouth, E. O. Shaughnessy, and S. Forrester, "Tracking the Sun Systems in the United States 2021 Edition," no. September, 2021.

[2] M. R. Haq and Z. Ni, "Classification of electricity load profile data and the prediction of load demand variability," *IEEE International Conference on Electro Information Technology*, vol. 2019-May, pp. 304–309, 2019.

TABLE III: A comparison of the features of the proposed RM updating method

Method	Advantages	Disadvantages	Applications
Additive	• Lossless SP values • Simple • Preserves data	• Memory intensive • Highest time complexity	• Long-term behavioural studies with SP
Fixed-memory	• Lowest memory requirement • Fast update (low-inertia) • Feature enhancement (high-inertia) • Balanced performance (medium-inertia)	• Low accuracy • Permanent knowledge elimination (in low-inertia strategy)	• Short-term modelling • Short-term tracking of changes
Codebook-based	• Low memory requirement • Preserve the changes from historical records • Flexible • High security (no sequential information) • High compression ratio for long TS	• Cannot preserve original data • Memory intensive for short TS	• Long-term users • Sensitive data to preserve privacy

[3] B. P. Butunoi and M. Frincu, "Shapelet based classification of customer consumption patterns," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2017 - Proceedings*, vol. 2018-Janua, 2017, pp. 1–6.

[4] S. M. Bidoki, N. Mahmoudi-Kohan, M. H. Sadreddini, M. Z. Jahromi, and M. P. Moghaddam, "Evaluating different clustering techniques for electricity customer classification," *2010 IEEE PES Transmission and Distribution Conference and Exposition: Smart Solutions for a Changing World*, pp. 1–5, 2010.

[5] Q. Wang and V. Megalooikonomou, "A dimensionality reduction technique for efficient time series similarity analysis," *Information Systems*, vol. 33, no. 1, pp. 115–132, 2008.

[6] R. Yuan, S. A. Pourmousavi, W. L. Soong, G. Nguyen, and J. A. Liisberg, "Irmac: Interpretable refined motifs in binary classification for smart grid applications," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 11, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197622005784

[7] Y. N. Silva, S. S. Pearson, J. Chon, and R. Roberts, "Similarity joins: Their implementation and interactions with other database operators," *Information Systems*, vol. 52, pp. 149–162, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.is.2015.01.008

[8] G. De Francisci Morales and A. Gionis, "Streaming similarity self-join," *Proceedings of the VLDB Endowment*, vol. 9, no. 10, pp. 792–803, 2016.

[9] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 493–498.

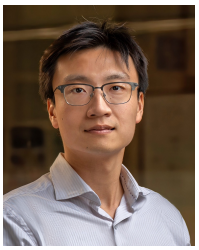
[10] C. C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 1317–1322, 2017.

[11] R. Yuan, S. A. Pourmousavi, W. L. Soong, A. J. Black, J. A. Liisberg, and J. Lemos-Vinasco, "Unleashing the benefits of smart grids by overcoming the challenges associated with low-resolution data," *Cell Reports Physical Science*, vol. 5, no. 2, p. 101830, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666386424000559

[12] Ausgrid. Solar home electricity data. [Online]. Available: https://www.ausgrid.com.au/Industry/Our-Research/Data-to-share/Solar-home-electricity-data

[13] A. G. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. O'Hare, "Real-time recognition and profiling of appliances through a single electricity sensor," *SECON 2010 - 2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2010.

[14] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? The case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.enpol.2012.08.062



Rui Yuan received the B.Sc. degree in Telecommunication Engineering from Harbin Institute of Technology, China, in 2016, the M.Sc. degree in Electrical Engineering from the University of Melbourne, Australia, in 2019, and the Ph.D. degree in Electrical and Electronic Engineering from the University of Adelaide, Australia, in 2025. He is currently a load forecasting analyst at AGL. His main research interest is the analysis and modelling of energy consumption profiles of consumers. This includes data mining, explainable machine learning, synthetic data generation and time series analysis.

S. Ali Pourmousavi (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees (with hons.) in electrical engineering, in 2005, 2008, and 2014, respectively. From 2014 to 2019, he was with California ISO, NEC Laboratories America Inc., Technical University of Denmark, Kongens Lyngby, Denmark, and The University of Queensland, Brisbane, Australia. He is currently a Senior Lecturer in the school of Electrical and Mechanical Engineering, the University of Adelaide, Australia. His current research interests include mining and transportation electrification, BTM flexibility aggregation and demand response.

Wen L. Soong (Fellow, IEEE) received the B.Eng. degree in electrical engineering from the University of Adelaide, Australia, in 1989, and the Ph.D. degree in electrical engineering from the University of Glasgow, Glasgow, U.K., in 1993. From1994 to 1998, he was with General Electric Corporate Research and Development, Schenectady, NY, USA. In 1998, he joined the University of Adelaide where is he is now an Associate Professor. His research interests include PM and reluctance machines, renewable energy generation, and energy storage in power systems.

Jon A. R. Liisberg received the B.Sc. degree in mathematics from the University of Copenhagen, Denmark, in 2013, and the M.Sc. and Ph.D. degrees in mathematical modelling and computation from the Technical University of Denmark in 2015 and 2019, respectively. His Ph.D. was jointly funded by Innovation Fund Denmark and Watts A/S. He is currently employed as a Senior Data Scientist with Watts A/S, with focus on developing and maintaining models and methods that enhance utility data and provide valuable insights, while also aiding the operations of the business.