IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 14, 2018, accepted December 5, 2018, date of publication December 20, 2018,
date of current version January 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2888923

SPADE: Activity Prediction in Smart Homes Using Prefix Tree Based Context Generation

ARAF FARAYEZ, MAMUN BIN IBNE REAZ, (Senior Member, IEEE), AND NORHANA ARSAD, (Member, IEEE)

Center of Advanced Electronic and Communication Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, 43600 Bangi, Malaysia

Corresponding author: Araf Farayez (araf.farayez@gmail.com)

This work was supported by Universiti Kebangsaan Malaysia under Grant GUP-2017-059.

ABSTRACT An automated smart home system is a key contributor to user assistance technology in modern civilization. Crucial merit of such a system is its ability to train itself through recorded data and recognize patterns in resident behaviors. Lack of sufficient prediction accuracy, exponential memory consumption, and extensive runtime prevent many of the current activity prediction approaches from being seamlessly integrated into consumer residences. This research introduces a sequence prediction algorithm which uses a prefix tree-based data model in order to learn and predict user actions. The algorithm applies episode discovery to detect correlated sensor events and learns the activities using a lossless data compression technique. This process assigns a probability of occurrence to sensor events and uses these probabilities to detect patterns in resident behavior. A complexity analysis of the algorithm is done to prove its efficiency in terms of memory usage and runtime. Using the presented technique, predictions are performed on popular datasets and contrasted with existing algorithms. The proposed algorithm achieves an 8.22% improvement in prediction accuracy over its predecessors, along with 66.69% better memory efficiency and 37% faster runtime.

INDEX TERMS Activity prediction, data compression, prediction by partial matching, prefix tree, sequence prediction, smart home.

I. INTRODUCTION

In this era of an indoor generation where humans are spending a great deal of time within their homes more than ever, smart home or automated home system is becoming a daily necessity. It immensely helps people who are handicapped or has a hard time regulating within their households, as well as serves the elderly citizens of a specific community, by making their lives easier to maintain. The Smart Home Association defines a smart home as ''an integration of technology and services through home networking for a better quality of living'' [1]. Such services are set up through predicting individual's daily activities and it is done by collecting ADL (activities of daily living) data. Performance of an automated home almost entirely depends on the system's observation, perception, and effective usage of prediction algorithms. A proper integration requires the system to observe an individual's activity pattern and formulate a probable next event based on the learned data.

A diverse variety of living smart home researches are conducted around the world. Adaptive House in USA [2], MavHome [3], [4], House of the Future or House-n in MIT [5], and Aware Home Research Initiative (AHRI) at Georgia Tech [6] are the most notable research initiatives for smart environment research.

Previously, automated home systems focused on appliance intelligence more than human behavior. But for a developed scheme it requires close observation, being able to predict the next course of actions and act out on the best decisions developed by the system itself. This is achieved by learning their behavior and further analyzing them using an existing computing framework. With the contribution of numerous researchers, several activity prediction systems are developed over the past few decades [7]. Artificial neural networks [8], Bayesian statistics [9], fuzzy logic [10] are some of the approaches used to provide automation in smart homes.

PUBS (Patterns of User Behavior System) is an activity prediction approach developed by Asier et al. This method uses three algorithms to identify resident actions, learn those actions, and interact with users. L_{PUBS} algorithm is used to correlate resident actions, define rules, and identify patterns. A_{PUBS} is used to learn activity patterns through the

established rules. And finally, I_{PUBS} is the interface that users can interact with and fine-tune their preferences [11].

A more recent study implements the Back Propagation Neural Network (BPNN) in order to predict resident actions in smart homes. Xu *et al.* [12] modified the BPNN to develop Parallel-BPNN that runs more efficiently for larger data quantity. In this method, a Neural network is trained with smart home sensor data by adjusting weights of the network using backpropagation.

A study done by Erfaneh et al. is noteworthy for their focus on the variations of living conditions of 5 different target groups. The authors developed a 3D interactive smart space using virtual reality (VR) in order to collect residents' behavioral data. This is particularly effective as the stated preferences of residents are collected and habits of each individual are recorded. The living patterns are predicted using the Bayesian Belief Network on the basis of individual data models [13].

Recently, data compression techniques gained enormous recognition among smart home researchers due to their pattern predicting capabilities [14]. Bhattacharya and Das developed LeZi update algorithm by implementing LZ78. It is an effective data compression algorithm which uses dictionary contexts to develop a user behavior model [15]. Although the LeZi update can predict user actions fairly accurately, it has a rather slow convergence rate, making this algorithm unfeasible in real-world applications [16]. Gopalaratnam and Cook introduced a variable length window to solve this problem and developed Active LeZi (ALZ) [17]. In this process, more weight is associated with recent activities, which makes this algorithm more sensitive towards the latest data. ALZ is further developed by Vikramaditya and Cook by adding temporal rules, which made this algorithm more responsive to the timestamp of user actions [18].

Prediction by Partial Matching (PPM) is another data compression method used by Alam *et al.* [19] to develop SPEED (Sequence Prediction via Enhance Episode Discovery). In this approach user activities are classified using distinct episodes consisting of sequential events. The classified episodes are collected in a finite order Markov model. An average prediction accuracy of 88.3% is achieved which surpasses the performance of LeZi Update and ALZ algorithm. Further development of SPEED is performed by Marufuzzaman *et al.* who included the time and location components to formulate Modified-SPEED or M-SPEED [20], [21]. Addition of time component can reduce redundancy in data caused by corrupted sensors. This, in turn, increases the accuracy of prediction. On the other hand, with the help of location component, the SPEED can be more efficient in classifying sensors based on their location which in turns achieves a higher reliability in prediction. This is mainly important in scenarios where sensor locations can play a vital role in user activity patterns.

A major drawback of the SPEED lies in its process of training the data model by tree generation. Due to the use of a linear approach in the tree formation, different branches of the tree often have misaligned probabilities. This phenomenon is further discussed in the later section of this paper. As a result of this inconsistency, the algorithm is heavily dependent on the structure of its training data. In this paper, we introduced Sequence Prediction via All Discoverable Episodes (SPADE), that improves over the novel SPEED algorithm. The proposed algorithm introduces a more effective way to handle the tree generation in order to address the limitations of SPEED. At the same time, this paper puts forward a few enhancements that can lead to a higher accuracy and better performance than the previous designs.

II. METHODOLOGY

The proposed SPADE algorithm takes the raw data from the training dataset and processes the data in several steps in order to generate a trained model. Main 4 steps of this procedure are discussed in this section.

A. SEQUENCE CREATION

In the raw training data, each user activities are recorded in separate lines. Different data formats are utilized by various datasets. Sequence Creation Modules differ in the parsing process based on the structure of this raw data. Proposed research chiefly uses CASAS *adlnormal* dataset [22] and MavLab dataset [23] for predicting user activities.

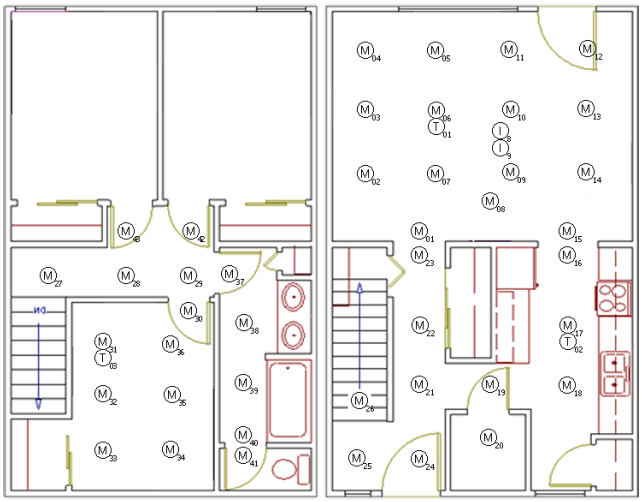

FIGURE 1. Sensor locations around the rooms in Kyoto testbed [22].

The CASAS *adlnormal* dataset is collected in the Kyoto testbed, which is a three-bedroom apartment in the WSU University Housing system [22]. The dataset contains 6425 data points gathered for 20 healthy adult test subjects. There are in total of 26 motions sensors distributed around the house. Item sensors and equipment sensors are also used to record specific user actions. The positions of the sensors inside rooms are depicted in Fig. 1 and positions of the sensors inside the cabinet are shown in Fig. 2. The residents performed 5 predetermined activities in no particular order. The ADLs consist of – making a phone call, washing hands, cooking, eating,

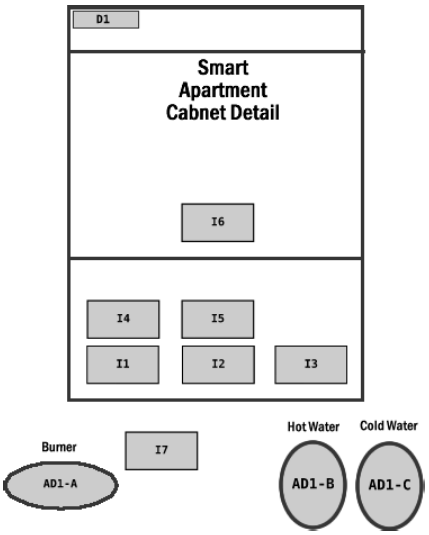

FIGURE 2. Sensor locations inside the cabinet of Kyoto testbed.

TABLE 1. Data format in different datasets.

Dataset	Data Format	Value Separator
CASAS adlnormal	[Date (yyyy-mm-dd)] [Time (hh:mm:ss.ms)] [Sensor ID] [Sensor State] [Activity Name (Optional)] [Activity Value (Optional)]	Tab Character
MavLab	[Date (dd/mm/yyyy)] [Time (hh:mm)] [Room ID] [Sensor ID] [Sensor State]	Space Character

and cleaning. The tasks are recorded separately and no overlapping activities are performed by the residents. Thus, it is a single resident database. Individual behaviors and habits of test subjects are omitted in order to generalize the activity pattern.

Another dataset used in this research is the MavLab dataset collected in the MavHome testbed of University of Texas, Arlington. In this collection, activities of 6 inhabitants are recorded over a period of one month. It is a single resident database similar to the CASAS *adlnormal*. The testbed comprises of 50 different appliances and in total around 700 valid sequential states are recorded. The residents performed day to day activities without any guidance. This resulted in a realistic situation for single resident inhabited homes [3], [23].

It is to be noted that, while the CASAS dataset has more data points than the MavLab, the number of appliances is significantly higher in the case of MavLab dataset. Data formats of these datasets are presented in Table 1.

A 5-step approach is taken to process the raw data and make it suitable for the algorithm. At the end of this process, a linear sequence of activities is compiled and sent to the next module for episode extraction.

1) STEP 1

Any unwanted data is removed in this step. False sensor readings, invalid rows in the dataset, and blank spaces are managed. Additionally, for the purpose of this research, only Motion sensor readings are extracted.

2) STEP 2

In this step, the raw data is parsed from the dataset and stored in a 2-dimensional array. In case of CASAS *adlnormal* dataset, the records are classified based on Person ID found in file names.

3) STEP 3

Time is converted to a 5-digit string using the formula: hour*3600 + minute*60 + second. For the purpose of this research, the millisecond precision of the time is ignored as it does not contribute to the accuracy of the prediction.

4) STEP 4

The Sensor IDs in the raw data are generally denoted by several characters. For example: 'M02' may indicate the 2nd Motion sensor. In this step, these values are mapped to ASCII characters in order to denote by one character. The Sensor States are merged with Sensor IDs thus allocating only once character to determine sensor name and its state. During this mapping 2 constants: *START*, and *DIVIDER* are introduced to denote the starting points of ON to OFF states. A variable *OFFSET* can be used to convert from ON to OFF state and vice versa where,

$$OFFSET = DIVIDER - START \tag{1}$$

In this research, values of *START* and *DIVIDER* are set as 33 and 80 respectively. This way, any whitespace characters can be avoided from the calculation, while using the maximum number of consecutive ASCII characters. The maximum number of sensors denoted in this approach is equal to *OFFSET*. For the specified values of *START* and *DIVIDER*, the maximum number of available sensors is 47. Sensor mapping for Person 1 of CASAS *adlnormal* database is shown in Table 2.

TABLE 2. An instance of sensor mapping.

Sensor ID in Raw Data	New Sensor ID (ASCII Character)	New Sensor ID (Value)
'M01'	'!'	33
'M07'	'"'	34
'M08'	'#'	35
'M09'	'$'	36
'M13'	'%'	37
'M14'	'&'	38
'M15'	'''	39
'M16'	'('	40
'M17'	')'	41
'M18'	'*'	42
'M23'	'+'	43

In the case of the MavLab dataset, the room number is also included in the newly generated sensor codes. Here, the approach developed by Marufuzzaman *et al.* [21] is used which uses the following equation:

$$newSensor = (room - 1) \times MR + rawSensor \tag{2}$$

where,
newSensor = Modified sensor ID
room = Room number
MR = Maximum number of sensors in one room
rawSensor = Sensor ID in raw data.

5) STEP 5

Finally, the processed values are compiled in one continuous sequence of records. The sequence is stored in a 2-dimensional array in the format: **[Sensor Code] [Date] [Time]**

While using ASCII symbols renders effective computation, human interpretation of these symbols is exceedingly hard. For a clear understanding of this paper in the following sections, the events are represented using Uppercase and Lowercase letters. Uppercase representing ON state of an event and Lowercase representing OFF state. For example, in any event sequence 'ABcabC', 'A' corresponds to ON state of a specific sensor whereas 'a' corresponds to the OFF state of the same sensor.

B. EXTRACTION OF EPISODES

Episodes are extracted following the windowing technique used in SPEED algorithm. A variable length window is used to accumulate episodes from the generated sequence. A different approach is taken in this research that exploits the advanced matrix manipulation capabilities offered by MatLab to generate the same resultant episodes more efficiently. The pseudocode of Episode Generation Module as shown in Fig. 3 eliminates the need for a nested loop to search for subsequent events constituting the episodes.

```
Extract episodes for sequence Seq with time T:
initialize storage Ep to an empty array

loop for every event V in Seq

  if event V is in ON state
    OFF_V := first OFF state of V after current
event
    ON_V := all ON states of V between V and OFF_V

    Time_ON := T(ON_V)
    Time_ON_max := MAX(Time_ON)
    Time_OFF := T(OFF_V)

    if Time_OFF>Time_ON_max
      S := events between V and OFF_V
      remove duplicates of V from S
      store S in storage Ep
    end
  end

end loop
```

FIGURE 3. Pseudocode for episode extraction.

During collection of episodes, the time of events is used to validate the integrity of sensors. For episodes where event E_1 comes before E_2 but $Time(E_1) > Time(E_2)$, an invalid sensor firing is identified and the episode is omitted. Additionally, the duplicate firings of same sensor event are removed to better facilitate the subsequent modules.

C. GENERATION OF ALL POSSIBLE CONTEXTS

In this step, the episodes are processed to generate all possible contexts to build the data tree. In SPEED an intuitive linear strategy was used to get the possible combinations while keeping the order of events intact [19]. In case of an episode 'Bcdb', the algorithm generated combinations: 'B', 'c', 'd', 'b', 'Bc', 'cd', 'db', 'Bcd', 'cdb', and 'Bcdb'. These combinations were then stored in linear arrays to be used in the later part of the algorithm. But this method may result in multiple memory allocations for the same combination in different episodes. For a better understanding let us consider another episode 'Ecd' occurring after 'Bcdb'. For 'Ecd' the generated combinations are 'E', 'c', 'd', 'Ec', 'cd', and 'Ecd'. Here, combinations 'c', 'd', and 'cd' are repeated and will require additional memory to process.

To resolve the above-mentioned problem, a prefix tree-based approach is introduced in this research. Instead of using a linear array to store generated possibilities, a prefix tree (also known as Trie tree) [24] data structure is implemented that can grow on demand and simultaneously record the frequency of occurrences of the possibilities. Singly connected linked lists are used to connect nodes in the tree. A root node σ is connected to multiple subtrees where node T_x occurring as a child of node T_y indicates that event T_y is preceded by event T_x in the training dataset.

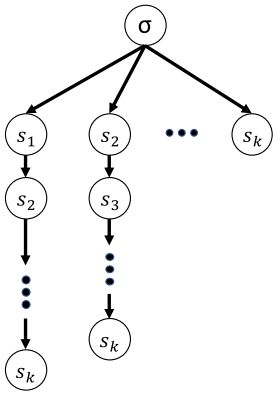

FIGURE 4. A general structure of the data subtree.

Fig. 4 depicts the generalized structure of the possibility tree for an episode s of length k. Here, s_1, s_2,... s_n are events of episode s, where event s_n is followed by event s_{n-1} in the dataset. Thus, node s_n is added to s_{n-1} as a child node. If there is a repetition during possibility generation, a node will preexist in the tree representing the repeating combination. During this scenario, no new node is added, only the frequency of the existing node is incremented by a value of 1.

A step by step processing of an episode 'Bcdb' is visualized in Fig. 5. In this example, a root node 'σ' is first created as the origin of the decision tree. During the first iteration, event 'B' is added as a child node to the root node and its frequency is set as 1. During iteration 2, 'c' is added as a child to all its preceding events in episode 'Bcdb'. Thus, a child is added to root node 'σ' and its child 'B'. Concurrently,

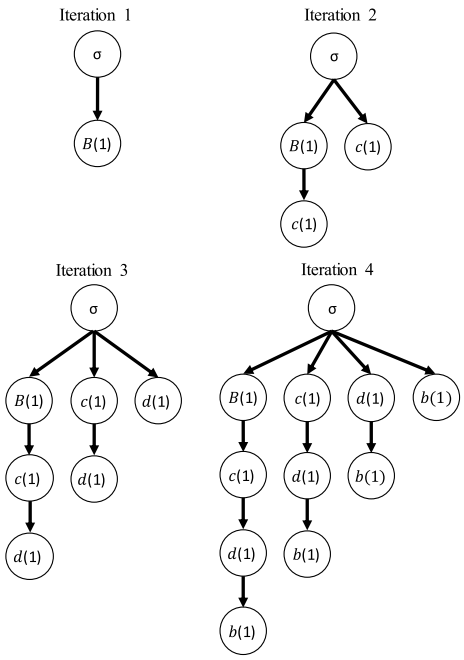

FIGURE 5. Tree generation steps for episode *'Bcdb'*.

```
Generate all possible contexts for episodes in
storage Ep:
initialize root of the tree Root to a blank node

loop for every episode S in Ep
  initialize tracker L to an empty array (used to
track subtree nodes)

  loop for every event V in episode S
    loop for every node N in tracker L
      if node N has a child node Nv with value V
        increment frequency of Nv by 1
      else
        add node Nv as a child to node N with value V
        initialize frequency of Nv to 1
      end
      replace N with Nv in tracker L
    end loop

    if Root has a child node Rv with value V
      increment frequency of Rv by 1
    else
      add node Rv as a child to node N with value V
      initialize frequency of Rv to 1
    end

    insert node Rv in tracker L

  end loop
end loop
```

FIGURE 6. Pseudocode for generating all possible contexts.

the frequency of 'B' is incremented by 1 and frequency of newly created 'c' nodes are set as 1. In a similar fashion, iteration 3 and 4 finishes the tree building for episode 'Bcdb'.

The algorithm uses pseudocode in Fig. 6 to generate the data tree from extracted episodes. The method uses an array L to keep track of the recently added leaf nodes. In every iteration, the subtrees are added only to the leaf nodes contained in L and the new leaves are replaced in the array as new tracked items. This technique eliminates the need for multiple nested loops and exhaustive tree search in order to convert the linear episodes into prefix tree data structure.

Aside from the considerable improvement in memory and time complexity in the context generation phase, this method eliminated the need for additional tree generation phase required in the previous SPEED algorithms.

D. PROBABILITY ASSIGNMENT

In the previous SPEED algorithms, the Tree generation step following the All possible context generation module was used to simultaneously create the decision tree and allocate the probability of outcomes. This resulted in misaligned probabilities among different branches of the tree. A primary goal of this research is to separate the tree generation from the probability allocation. This technique can ensure uniform distribution of probabilities and significantly improve the accuracy of prediction.

The decision tree formed during the Possibility generation phase is traversed using a Depth First Search (DFS) traversal to visit all nodes and set their respective probabilities. At any given level the algorithm first goes to the leftmost child node and recursively reaches its bottom. Upon reaching a leaf node, sets its probability value and travels back to its immediate parent node. When the leftmost node child node is already traveled, the algorithm steps to the child node to its right and follows the same recursive procedure. This way every subtree is visited from bottom to top and left to right. An overview of this process for a particular training scenario is shown in Fig. 7. In this setting, the algorithm first goes to the leftmost child node of the root and travels to its bottom. When the node *D(1)* is reached, the algorithm sets the probability using the following equation:

$$p(s_n) = f(s_n)/f(s_{n-1}) \tag{3}$$

where,

$p(s_n)$ = Probability of node s_n

$f(s_n)$ = Frequency of node s_n

and, s_n is the child node of s_{n-1}.

Upon setting the probability of node *D(1)* to 0.5, its parent node is visited and the same procedure is repeated. After successive 4 iterations, the probability of all the nodes in the leftmost subtree is set and the algorithm moves to the subtree of node *b(4)*. This way the whole data tree is configured to its accurate probability in O(n) complexity.

III. COMPLEXITY ANALYSIS

The modifications of the SPADE algorithm in Possibility generation module improves the memory efficiency significantly. The following 4 hypotheses are presented to analyze the memory complexity of this algorithm and establish the effectiveness of the proposed changes.

Hypothesis 1: In the worst-case scenario, M-SPEED has a space complexity of k to the power of three for an episode of length k.

Proof: In the All Possible Context Generation module, the generated contexts are stored in a 2-dimensional linear

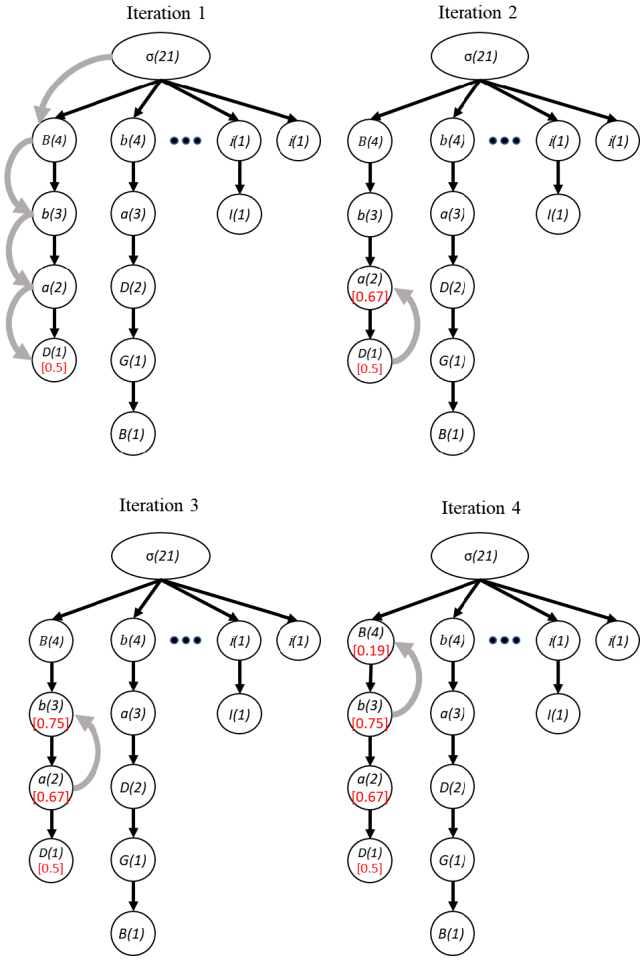

FIGURE 7. Probability assignment steps for a particular training scenario.

array to create the tree. For any episode s of length k, the number of possible contexts with length t will be $k - t + 1$.

To demonstrate this, let us consider an episode 'Bcdb'.

- Possibilities containing 1 letter are 'B', 'c', 'd', and 'b' (in total 4).
- Possibilities containing 2 letters are 'Bc', 'cd', and 'db' (in total 3).
- Possibilities containing 3 letters are 'Bcd' and 'cdb' (in total 2).
- Possibility containing 4 letters is 'Bcdb' (in total 1).

Therefore, the total amount of required memory (letters) for an episode of length k is,

$$M_1(k) = \sum_{t=1}^{k} t * (k - t + 1) = \frac{k^3 + 3k^2 + 2k}{6} \tag{4}$$

Hence, the worst-case space complexity is $O(k^3)$, where k is the number of events in an episode.

Hypothesis 2: In M-SPEED, the worst-case scenario appears when the complete data consists of a single large episode and no events are repeated in that episode.

Proof: For any sequence of events D that consists of n number of episodes of length k_n, where $k = \sum k_n$, we know,

$$k^3 \geq \sum (k_n)^3 \quad \text{And,} \tag{5a}$$

$$k^2 \geq \sum (k_n)^2 \tag{5b}$$

Therefore, based on the values in Equation (1) we can infer,

$$M_1(k) \geq M_1(k_n) \tag{6}$$

This implies that, when the whole dataset contains only one episode, the algorithm allocates the most memory.

Hypothesis 3: In the worst-case scenario, SPADE has a space complexity of k to the power of two for an episode of length k.

Proof: This research implements a Prefix Tree based model where nodes are connected using linked lists. Every node contains one letter (to denote its event), frequency, and an address to the next node. For an episode s of length k, the number of nodes at level l will be $k - l + 1$, where $1 \leq l \leq k$.

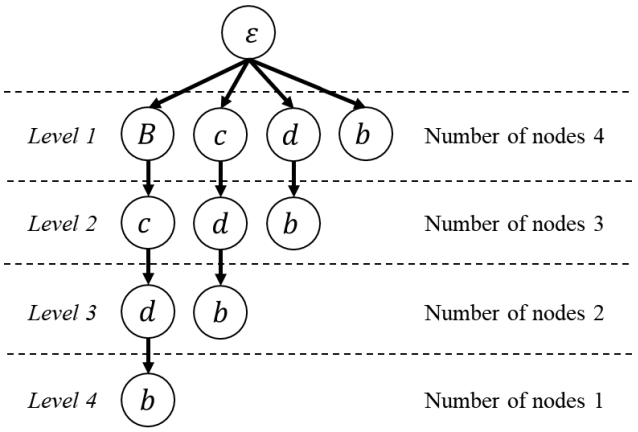

FIGURE 8. Context tree formed for episode 'Bcdb'.

This argument can be demonstrated by visualizing the tree formed by the episode 'Bcdb' in Fig. 8. In this example, for each level, the node count is decremented by 1.

Therefore, for an episode of length k, the sum of required memory is:

$$M_2(k) = \sum_{l=1}^{k} (k - l + 1) = \frac{k(k+1)}{2} \tag{7}$$

Hence, the worst-case space complexity of the SPADE algorithm is in the order of $O(k^2)$, where k is the number of events in an episode.

Hypothesis 4: In SPADE, the worst-case scenario appears when the complete data consists of a single large episode and no events are repeated in that episode.

Proof: Similar to Hypothesis 2, when a data sequence is divided into n number of episodes of length k_n, where $k = \sum k_n$, we can say,

$$k^2 \geq \sum (k_n)^2 \tag{5b}$$

Putting these values in Equation (4) we find,

$$M_2(k) \geq M_2(k_n) \tag{8}$$

This proves that dataset having one large episode covering the whole sequence will require the maximum amount of memory.

IV. RESULTS AND DISCUSSIONS

The SPADE algorithm is designed to yield better accuracy and performance compared to the existing activity prediction algorithms like SPEED, M-SPEED, PUBS, and BPNN. To validate our proposal, we used two popular datasets CASAS *adlnormal,* and MavLab to run different algorithms and compare their results. Both these datasets are described in section II.A.

Firstly, SPADE, M-SPEED, and PUBS algorithms are trained and tested with the MavLab dataset to compare their peak accuracies over different training lengths. Secondly, the average accuracies of SPADE and M-SPEED for different episode lengths are compared to analyze the influence of resident behavior. Thirdly, the memory allocation and runtime of both these algorithms are recorded and compared in order to gain a better insight into how much performance improvement is possible with the current changes. Finally, SPADE is run on CASAS *adlnormal* dataset and compared with BPNN to assess the prediction accuracy for massive data points.

FIGURE 9. Accuracy comparison of SPADE, M-SPEED and PUBS on MavLab dataset for different training data sizes.

The first analysis of the performance of the SPADE algorithm is done in terms of peak accuracy over varying sizes of training data. M-SPEED and PUBS algorithms are implemented to compare their accuracies with this research. In Fig. 9 the prediction accuracies of all the three algorithms are plotted over increasing training lengths. To minimize the runtime, an interval of 20 is chosen to collect data points. From the experiment results, a steep increase is found in SPADE and PUBS for smaller data sizes, but when the number of training events exceeded 80, SPADE showed a significant improvement over the other two algorithms. The maximum accuracy is achieved at event size 220 and a stable performance is visible following that.

Fig. 10 charts the accuracy of SPADE and M-SPEED on MavLab dataset with respect to episode lengths. Here, the episode lengths refer to the number of sequential events constituting a particular activity. An accuracy of 78% for episode length 4 indicates that, when the trained model predicts activity after 3 consecutive known events, an average of 78% accuracy is achieved. The graph shows a uniform increase in prediction capability of around 10% for the presented approach. This is a concrete establishment that the proposed algorithm performs better than the previous M-SPEED in every possible episode length.

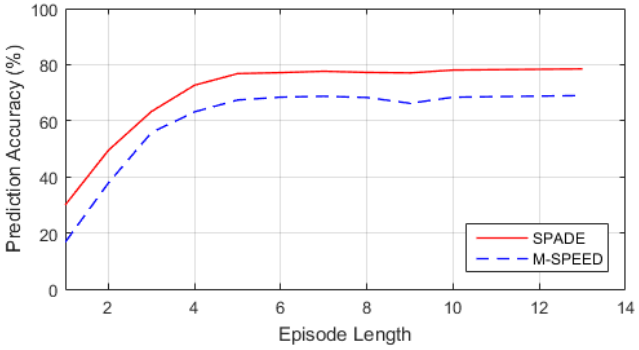

FIGURE 10. Accuracy comparison of SPADE and M-SPEED on MavLab dataset with time verification.

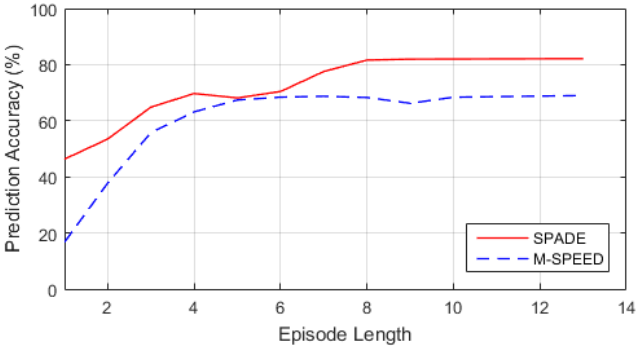

FIGURE 11. Accuracy comparison of SPADE and M-SPEED on MavLab dataset without time verification.

The importance of verification of event sequence based on time information is evident in Fig. 11. This figure shows the accuracy of SPADE without time verification. In this graph, thought the peak accuracy is higher than the proposed approach, anomalies are visible for episode length 5. Two major factors may play a role behind this anomaly. Firstly, the excessive random event firing in the dataset can cause random noise in the data. The mean of this noise can peak at a particular episode length causing the algorithm to render poor prediction for that length. Secondly, the lack of sufficient data points for a particular episode length can result in

false predictions. Nevertheless, the average accuracy is still higher than that of M-SPEED and its predecessors.

Better memory allocation is a key factor in determining the performance of any simulation. It can ensure a smooth transition of the process to hardware-based systems and assist the system to scale up for larger data pools. This concept was kept in mind during the development of the proposed algorithm. As mentioned earlier, the substitution of the linear array with a tree data structure can greatly conserve memory while delivering the same prediction accuracy. To prove this theory, memory allocation is recorded during runtime of both M-SPEED and SPADE for MavLab database. During the process, only byte counts for event tags are considered, due to the fact that frequency and time stay the same for both cases. Fig. 12 shows the average memory allocation with respect to different episode lengths. Whereas for small episode lengths the allocated memory is similar in both processes, a pronounced difference is visible for larger episodes. The overall effect on memory can be perceived in Fig. 13 where the cumulative memory growth of the simulations over successive episodes is displayed. As predicted, the exponential increase in array size is at play here. The trained data structure in SPADE is 66.69% more efficient than that of M-SPEED.

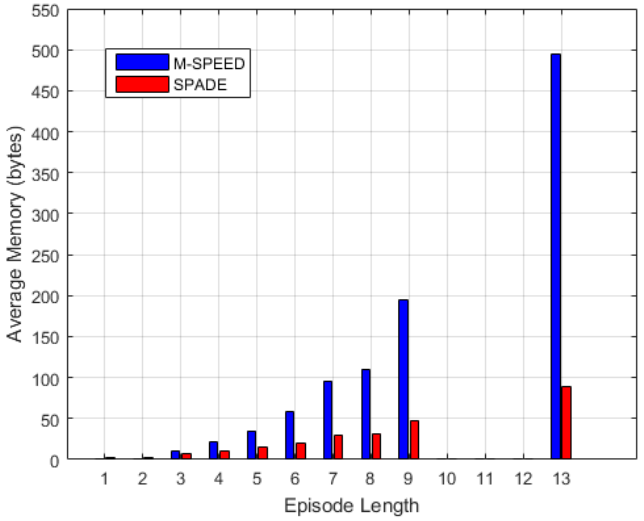

FIGURE 12. Average memory allocation of M-SPEED vs SPADE.

In addition to memory efficiency, SPADE also exhibits a reduced execution time compared to previous activity prediction systems. In M-SPEED, linear search on possibilities makes the process greatly dependent on data composition. A bigger size of the array causes an increase in the number of nested loops. Whereas the tree traversal method in SPADE is inherently faster, tree building is not. The use of tracker array discussed in section II.C can make tree building a lot faster and can process data in half the time. To prove this concept, MATLAB timing features are utilized in the simulation to record runtime. Fig. 14 demonstrates average time taken for every episode of the two contrasting algorithms. It is evident from the diagram that a notable improvement in performance

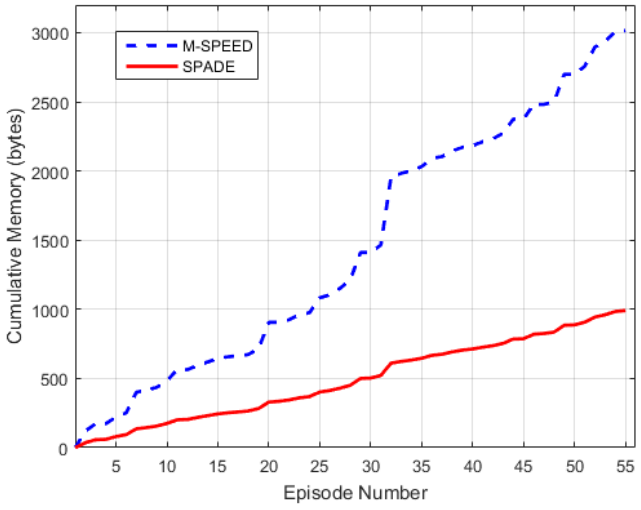

FIGURE 13. Cumulative memory allocation of M-SPEED vs SPADE.

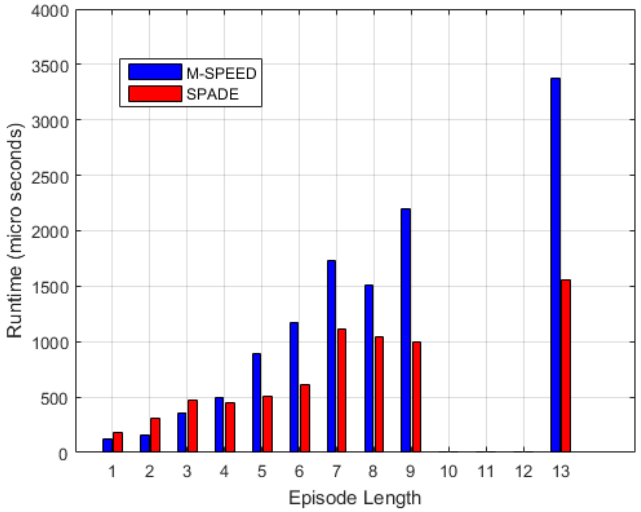

FIGURE 14. Average runtime of M-SPEED vs SPADE.

is found for larger episode lengths. For smaller episodes though, M-SPEED has a slighter advantage. This is due to the fact that, connecting nodes with linked lists require more function calls which in turn takes more time than linear loops. Overall, nearly 37% improvement in runtime is established by the new algorithm.

Because of the data management techniques applied to SPADE, this algorithm runs large datasets in polynomial time and memory. In the final experiment, we ran the SPADE algorithm on CASAS *adlnormal* dataset and compared the results with BPNN on the basis of training data sizes. It is to be noted that, due to a large number of data points in this dataset, M-SPEED and PUBS algorithms take a longer time to compute, and thus, omitted from this experiment. Fig. 15 shows the simulation results of SPADE and BPNN over increasing training events. Similar to the results on MavLab dataset, the performance of SPADE peaks earlier than BPNN and then maintains a stable prediction accuracy over the larger data sizes.

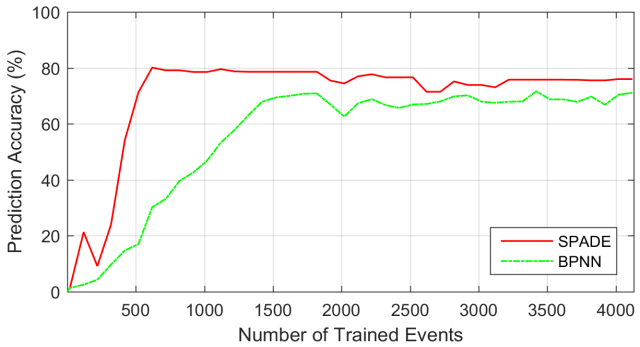

FIGURE 15. Accuracy comparison of BPNN and SPADE for different training data sizes.

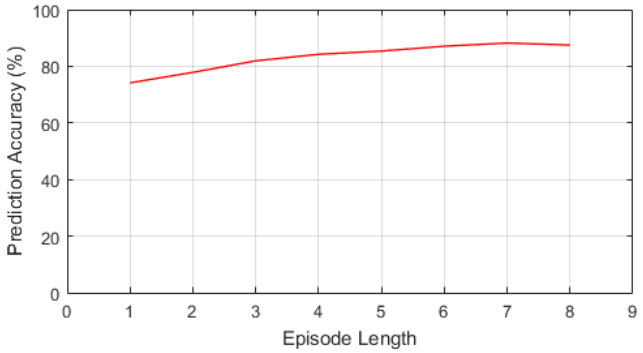

FIGURE 16. Accuracy of the proposed algorithm on CASAS dataset.

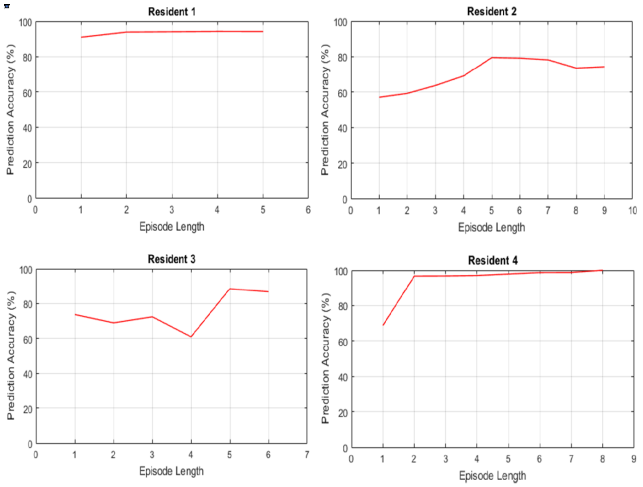

FIGURE 17. Accuracy of the proposed algorithm for individual residents of CASAS dataset.

From the graph in Fig. 16, we can see that the accuracies on CASAS *adlnormal* dataset for different episode lengths are remarkably higher than the tests run on the MavLab dataset. Even for predictions with no preceding events, the accuracy is more than 66%. The primary cause of this higher output is the abundance of data points in the CASAS dataset. Due to adequate training variety, the tree is built reliably and probability values are fine-tuned. Fig. 17 gives us an overview of 4 sample residents from the CASAS data pool. A closer inspection of these residents shows us that the prediction graph varies widely from person to person. This is attributed to the diversity in resident behavior and the inconsistency of the activity data. On one hand, resident 1 and 4 shows extremely high prediction values for their regularity in daily activities. On the other hand, resident 2 and 3 shows a wider variance, in some cases decline in accuracy with increasing episode lengths.

A brief summary of prediction accuracies of all the methods implemented in this research is given in Table 3. The accuracies over different data sizes are considered in this case. From this table it is evident that, at least 8.22% improvement is achieved by the proposed technique.

TABLE 3. Peak prediction accuracies of different algorithms.

Implemented Algorithms	Year	Peak Prediction Accuracy for CASAS *adlnormal* dataset (%)	Peak Prediciton Accuracy for MavLab dataset (%)
M-SPEED [21]	2017	--	72.53
PUBS [11]	2012	--	74.84
BPNN [12]	2017	72.25	--
This research	2018	80.47	84.11

Overall, the proposed SPADE algorithm has improved over the existing activity prediction algorithms in terms of both accuracy and computational complexity. The developed system can be applied to real-world scenarios and realistic datasets in order to deliver reliable activity prediction.

V. CONCLUSION

Recent smart home systems are equipped with the feature of activity prediction, which plays a crucial role to ensure proper automation and user comfort. The use of machine learning and data compression algorithms for predicting human activities is increasing rapidly in order to provide robustness to these systems. But with increasing data volume and high demand, these algorithms fail to deliver reliable accuracy and performance. In this paper, an improved human activity prediction algorithm is proposed based on a prefix tree-based data model. To ensure the viability of the proposal, simulations are conducted on popular datasets that closely resemble real-world user behavior. Results show a noteworthy improvement over previous approaches in terms of accuracy, memory management, and runtime. On the basis of these promising results, it is evident that our algorithm can be effectively used in smart homes applications to provide necessary automation.

REFERENCES

[1] J. R. Rosslin and K. Tai-Hoon, "Applications, systems and methods in smart home technology: A review," *Int. J. Adv. Sci. Technol.*, vol. 15, pp. 37–48, Nov. 2010.
[2] M. C. Mozer, "Lessons from an adaptive home," in *Smart Environments: Technologies, Protocols, and Applications*. New York, NY, USA: Wiley, 2005, pp. 271–294.
[3] D. J. Cook *et al.*, "MavHome: An agent-based smart home," in *Proc. 1st IEEE Int. Conf. Pervasive Comput. Commun.*, Mar. 2003, pp. 521–524.
[4] S. K. Das, D. J. Cook, A. Battacharya, E. O. Heierman, III, and T. Y. Lin, "The role of prediction algorithms in the MavHome smart home architecture," *IEEE Wireless Commun.*, vol. 9, no. 6, pp. 77–84, Dec. 2002.

[5] MIT House_n Researchers, "House_n research consortium," Massachusetts Inst. Technol., Cambridge, MA, USA, White Paper, 2005. [Online]. Available: http://web.mit.edu/cron/group/house_n/documents/HousenConsortiumResearchTopics.pdf
[6] J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd, "The georgia tech aware home," in *Proc. 26th Annu. CHI Conf. Extended Abstr. Hum. Factors Comput. Syst. (CHI)*, 2008, pp. 3675–3680.
[7] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A review of smart homes—Past, present, and future," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1190–1203, Nov. 2012.
[8] M. C. Mozer, "The neural network house: An environment that adapts to its inhabitants," in *Proc. Amer. Assoc. Artif. Intell. Spring Symp. Intell. Environ.*, Dec. 1998, pp. 110–114.
[9] C.-L. Wu and L.-C. Fu, "Design and realization of a framework for human–system interaction in smart homes," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 1, pp. 15–31, Jan. 2012.
[10] C. Anagnostopoulos and S. Hadjiefthymiades, "Advanced inference in situation-aware computing," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 39, no. 5, pp. 1108–1115, Sep. 2009.
[11] A. Aztiria, J. C. Augusto, R. Basagoiti, A. Izaguirre, and D. J. Cook, "Discovering frequent user–environment interactions in intelligent environments," *Pers. Ubiquitous Comput.*, vol. 16, no. 1, pp. 91–103, Jul. 2011.
[12] G. Xu, C. Shen, M. Liu, F. Zhang, and W. Shen, "A user behavior prediction model based on parallel neural network and k-nearest neighbor algorithms," *Cluster Comput.*, vol. 20, no. 2, pp. 1703–1715, 2017.
[13] E. Allameh, M. Heidari, and B. D. Vries, "Living preference modeling of smart homes for different target groups," *J. Ambient Intell. Smart Environ.*, vol. 10, no. 2, pp. 103–125, 2018.
[14] S. Wu *et al.*, "Survey on prediction algorithms in smart homes," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 636–644, Jun. 2017.
[15] A. Bhattacharya and S. K. Das, "LeZi-update: An information-theoretic framework for personal mobility tracking in PCS networks," *Wireless Netw.*, vol. 8, nos. 2–3, pp. 121–135, 2002.
[16] K. Gopalratnam and D. J. Cook, "Active Lezi: An incremental parsing algorithm for sequential prediction," *Int. J. Artif. Intell. Tools*, vol. 13, no. 4, pp. 917–929, 2004.
[17] K. Gopalratnam and D. J. Cook, "Online sequential prediction via incremental Parsing: The active LeZi algorithm," *IEEE Intell. Syst.*, vol. 22, no. 1, pp. 52–58, Jan. 2007.
[18] V. Jakkula and D. J. Cook, "Mining sensor data in smart environment for temporal activity prediction," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (Poster)*, 2007.
[19] M. R. Alam, M. B. I. Reaz, and M. A. Mohd Ali, "SPEED: An inhabitant activity prediction algorithm for smart homes," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 42, no. 4, pp. 985–990, 2012.
[20] M. Marufuzzaman, M. B. I. Reaz, M. A. M. Ali, and L. F. Rahman, "A time series based sequence prediction algorithm to detect activities of daily living in smart home," *Methods Inf. Med.*, vol. 54, no. 3, pp. 262–270, Jan. 2015.
[21] M. Marufuzzaman, M. B. I. Reaz, L. F. Rahman, and A. Farayez, "A location based sequence prediction algorithm for determining next activity in smart home," *J. Eng. Sci. Technol. Rev.*, vol. 10, no. 2, pp. 161–165, Jun. 2017.
[22] D. J. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods Inf. Med.*, vol. 48, no. 5, pp. 480–485, 2009.
[23] G. M. Youngblood and D. J. Cook, "Data mining for hierarchical model creation," *Appl. Rev.*, vol. 37, no. 4, pp. 1–12, 2007.
[24] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order Markov models," *J. Artif. Intell. Res.*, vol. 22, no. 1, pp. 385–421, 2004.

ARAF FARAYEZ was born in Dhaka, Bangladesh, in 1993. He received the B.Sc. degree in computer science and engineering from the Islamic University of Technology, in 2015. He is currently pursuing the M.Sc. degree with the Department of Electrical, Electronic and Systems Engineering, Universiti Kebangsaan Malaysia. His research interests include prediction algorithms, ubiquitous computing, and machine learning.

MAMUN BIN IBNE REAZ was born in Bangladesh, in 1963. He received the B.Sc. and M.Sc. degrees in applied physics and electronics both from the University of Rajshahi, Bangladesh, in 1985 and 1986, respectively, and the D.Eng. degree from Ibaraki University, Japan, in 2007. He has been a Senior Associate with the Abdus Salam International Centre for Theoretical Physics, Italy, since 2008. He is currently a Professor with the Department of Electrical, Electronic and Systems Engineering, Universiti Kebangsaan Malaysia, Malaysia, involving in teaching, research, and industrial consultation. He has vast research experiences in Japan, Italy, and Malaysia. He has published extensively in the area of IC design, biomedical application IC, and smart home. He has authored and co-authored over 300 research articles in design automation, IC design for biomedical applications, and smart home. He received over 50 research grants (national and international). He is a Senior Member of the IEEE.

NORHANA ARSAD (M'10) received the B.Eng. degree in computer and communication systems and the M.Sc. degree in photonics from Universiti Putra Malaysia, Malaysia, in 2000 and 2003, respectively, and the Ph.D. degree from the University of Strathclyde, Glasgow, U.K., in 2010. She is currently an Associate Professor with the Center of Advanced Electronic and Communication Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia. She is also active in engineering education and entrepreneurial. Her research interests include the investigation and design of fiber laser systems for application in spectroscopy, gas sensing, and photonics technology.