US 20240403648A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2024/0403648 A1

Xue et al. (43) Pub. Date: Dec. 5, 2024

(54) **INFERRING USER ACTIVITIES FROM INTERNET OF THINGS (IOT) CONNECTED DEVICE EVENTS USING MACHINE LEARNING BASED ALGORITHMS**

(71) Applicants: **Guoliang Xue**, Chandler, AZ (US); **Yinxin Wan**, Tempe, AZ (US); **Xuanli Lin**, Gilbert, AZ (US); **Kuai Xu**, Peoria, AZ (US); **Feng Wang**, Peoria, AZ (US)

(72) Inventors: **Guoliang Xue**, Chandler, AZ (US); **Yinxin Wan**, Tempe, AZ (US); **Xuanli Lin**, Gilbert, AZ (US); **Kuai Xu**, Peoria, AZ (US); **Feng Wang**, Peoria, AZ (US)

(73) Assignee: **Arizona Board of Regents on Behalf of Arizona State University**, Scottsdale, AZ (US)

(21) Appl. No.: **18/734,240**

(22) Filed: **Jun. 5, 2024**

Related U.S. Application Data

(60) Provisional application No. 63/506,316, filed on Jun. 5, 2023.

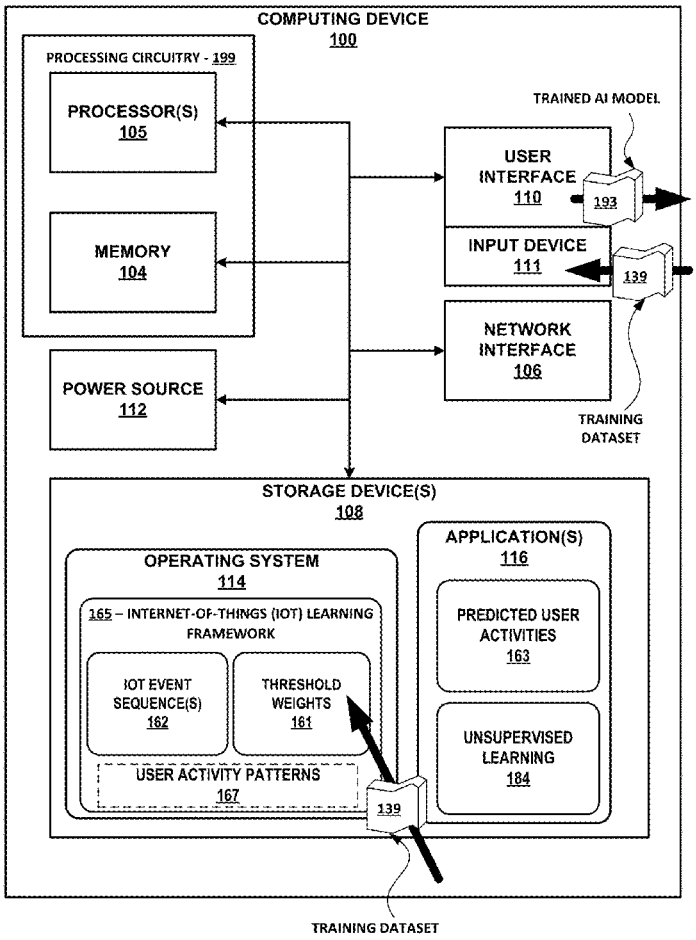
Publication Classification

(51) **Int. Cl.**
G06N 3/088 (2006.01)

(52) **U.S. Cl.**
CPC ***G06N 3/088*** (2013.01)

(57) **ABSTRACT**

An Internet-of-Things (IoT) learning framework (IoT learning framework) may train AI models to infer user activities from IoT connected device events using machine learning based algorithms. According to such an example, processing circuitry obtains a training dataset indicating IoT device events and extracts representative user activity patterns from the sequences of IoT device events. In such an example, processing circuitry trains the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold and outputs the AI model. According to such an example, processing circuitry may obtain new data indicating new sequences of IoT device events and generates output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.



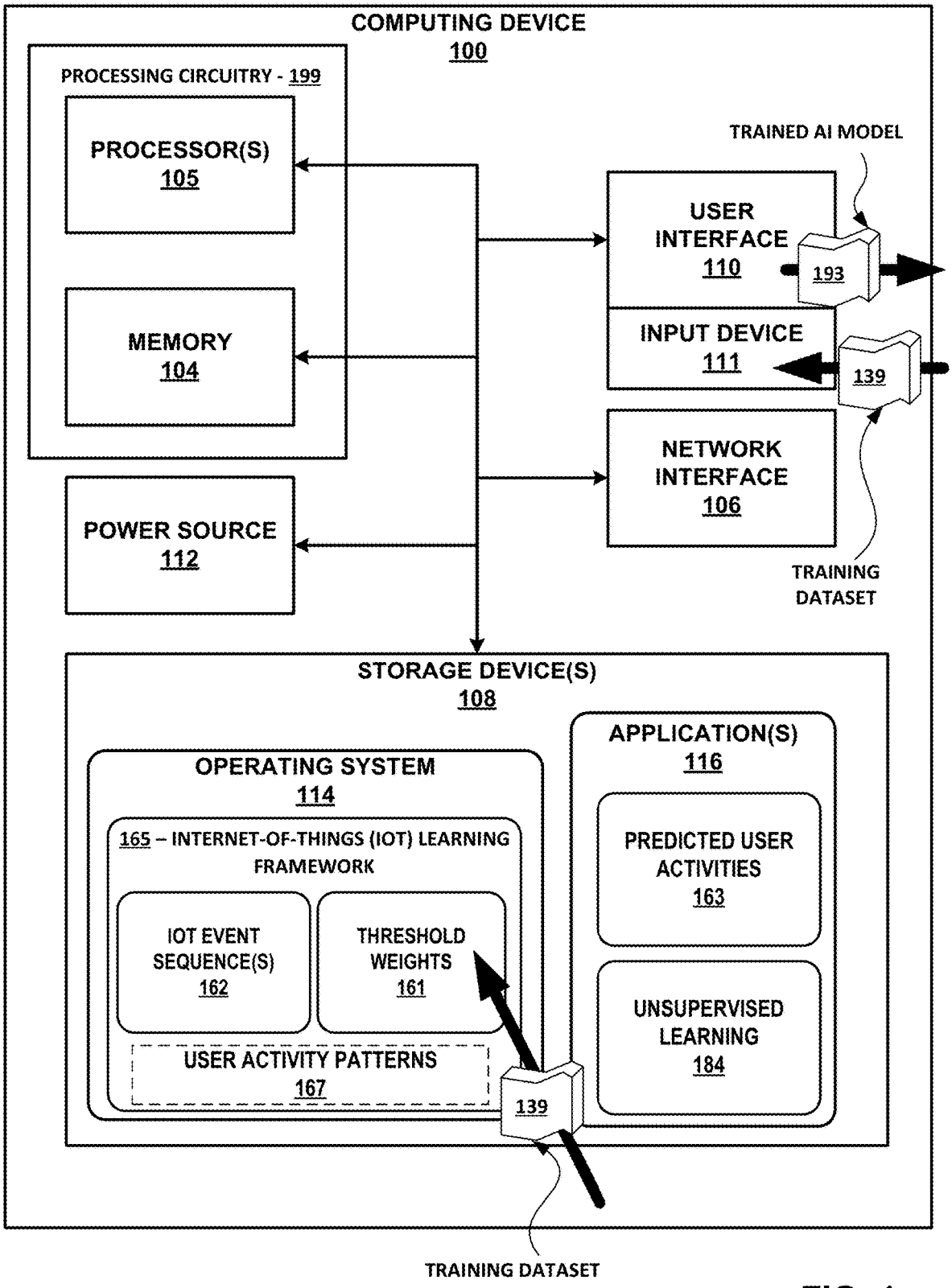

COMPUTING DEVICE
100
PROCESSING CIRCUITRY - 199
PROCESSOR(S)
105
MEMORY
104
POWER SOURCE
112
TRAINED AI MODEL
USER INTERFACE
110
193
INPUT DEVICE
111
139
NETWORK INTERFACE
106
TRAINING DATASET
STORAGE DEVICE(S)
108
OPERATING SYSTEM
114
165 – INTERNET-OF-THINGS (IOT) LEARNING FRAMEWORK
IOT EVENT SEQUENCE(S)
162
THRESHOLD WEIGHTS
161
USER ACTIVITY PATTERNS
167
139
APPLICATION(S)
116
PREDICTED USER ACTIVITIES
163
UNSUPERVISED LEARNING
184
TRAINING DATASET

FIG. 1

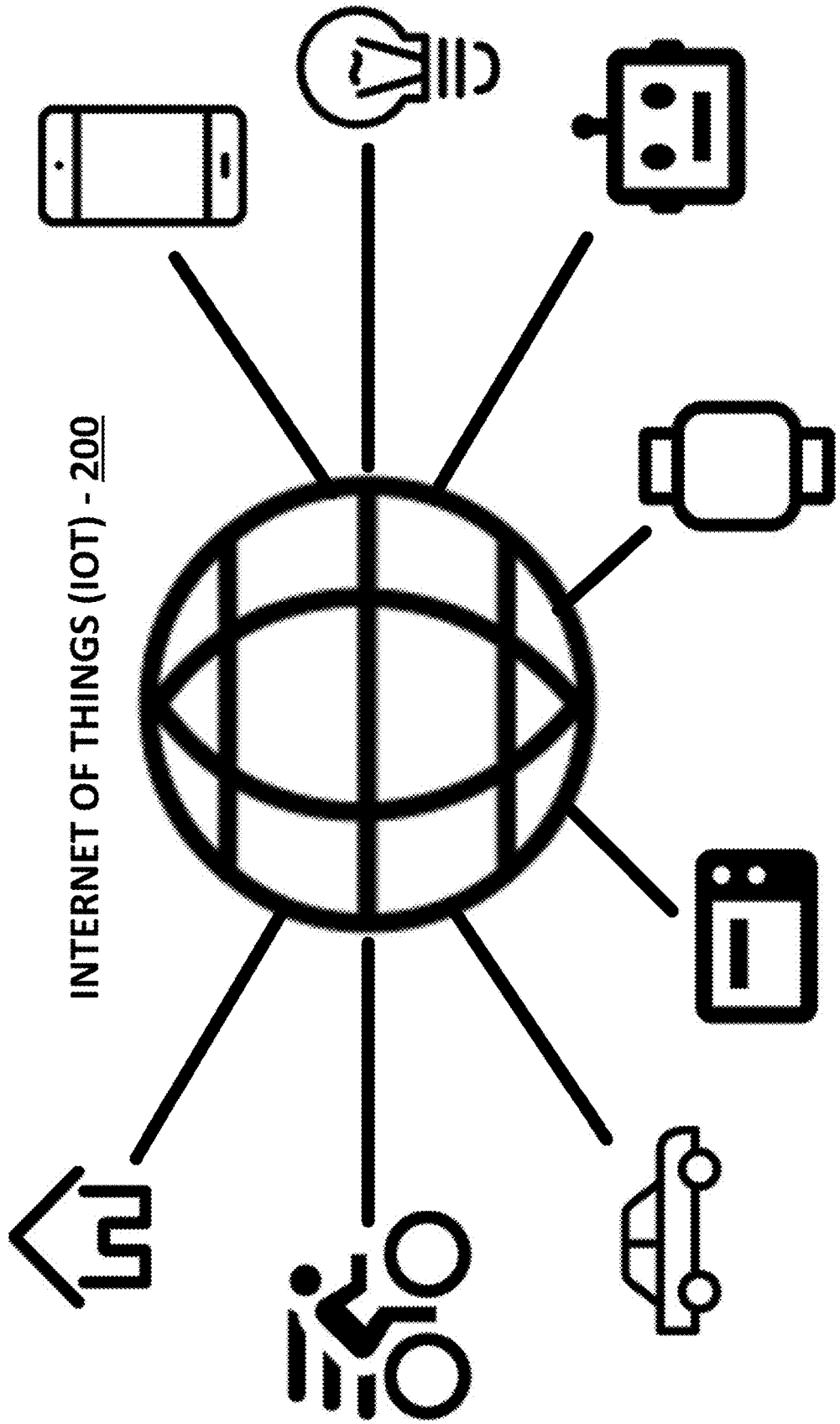


FIG. 2

301A – TABLE 1

NOTATIONS	DESCRIPTION
$\mathcal{A}$	set of user activities: $\mathcal{A} = \{A^1, A^2, \ldots, A^{\lvert\mathcal{A}\rvert}\}$
A^i	the ith user activity: $A^i \in \mathcal{A}$
A or A_j	a user activity: $A, A_j \in \mathcal{A}$
$\mathbb{A}$	sequence of user activities: $\mathbb{A} = (A_1, A_2, \ldots, A_{\lvert\mathbb{A}\rvert})$
$\mathbb{E}$	sequence of IoT device events: $\mathbb{E} = (e_1, e_2, \ldots, e_m)$
$\mathbb{E}(i:j)$	portion of $\mathbb{E}$ from index i to index j
$\mathcal{S}(A)$	set of distinct device event sequences triggered by $A \in \mathcal{A}$: $\mathcal{S} = \{\mathbb{S}^1(A), \mathbb{S}^2(A), \ldots, \mathbb{S}^{\lvert\mathcal{S}(A)\rvert}(A)\}$
$\mathbb{S}^k(A)$	the k-th possible device event sequence triggered by $A \in \mathcal{A}$: $\mathbb{S}^k(A) = (e_1^{A,k}, e_2^{A,k}, \ldots, e_{\lvert\mathbb{S}^k(A)\rvert}^{A,k})$

FIG. 3A

301B – TABLE 1 (continued)

NOTATIONS	DESCRIPTION
$\psi_i^{A,k}$	a match of $\mathbb{S}^k(A)$ in $\mathbb{E}$, with two interchangeable descriptions in the paper: $\psi_i^{A,k} = (\psi_{i,1}^{A,k}, \psi_{i,2}^{A,k}, \ldots, \psi_{i,\lvert\mathbb{S}^k(A)\rvert}^{A,k})$ or $\psi_i^{A,k} = (\psi_i^{A,k}[1], \psi_i^{A,k}[2], \ldots, \psi_i^{A,k}[\lvert\mathbb{S}^k(A)\rvert])$
$\phi_i^{A,k}$	similar to $\psi_i^{A,k}$
$\Psi^{A,k}$	set of all matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$
$\Psi^{min,A,k}$	set of all minimal matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$
Ψ	union of $\Psi^{A,k}$ over all combinations of $A \in \mathcal{A}$ and $k \in \{1, 2, \ldots, \lvert\mathcal{S}(A)\rvert\}$
Ψ^{min}	union of $\Psi^{min,A,k}$ over all combinations of $A \in \mathcal{A}$ and $k \in \{1, 2, \ldots, \lvert\mathcal{S}(A)\rvert\}$
$\mathbb{L}^{min,A,k}$	list of (not necessarily all) minimal matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$: whose i-th node may be denoted as $\mathbb{L}^{min,A,k}[i]$
$\mathbb{L}^{min}$	2-D array of lists of minimal matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$ for all possible combinations of $A^i \in \mathcal{A}$ and $k \in \{1, 2, \ldots, \lvert\mathbb{S}^k(A)\rvert\}$: indexed as $\mathbb{L}^{min}[i,k] = \mathbb{L}^{min,A^i,k}$

FIG. 3B

301C – TABLE 1 (continued)

NOTATIONS	DESCRIPTION
n_i	number of elements in $\mathbb{S}(A^i)$, $i = 1, 2, \ldots, \lvert\mathcal{A}\rvert$
N	$N = \sum_{i=1}^{\lvert\mathcal{A}} n_i$
$m_{i,k}$	$m_{i,k} = \lvert\mathbb{L}^{min,A^i,k}\rvert$, $i = 1, \ldots, \lvert\mathcal{A}\rvert$, $k = 1, \ldots, \lvert\mathbb{S}^k(A^i)\rvert$
M	$M = \sum_{i=1}^{\lvert\mathcal{A}\rvert} \sum_{k=1}^{\lvert\mathbb{S}^k(A^i)\rvert} m_{i,k}$
$\mathbb{M}$	1-D array of 4-tuple (A, k, α, β): $\mathbb{M}[i]$, $i = 1, 2, \ldots, M$
$w(i, k)$	weight for $\mathbb{S}^k(A^i)$, $i = 1, \ldots, \lvert\mathcal{A}\rvert$, $k = 1, \ldots, \lvert\mathcal{S}(A^i)\rvert$
$\lVert$	concatenation operator
$\mathbb{Z}^+$	set of nonnegative real numbers

FIG. 3C

401A – ALGORITHM 1

AkMatch($\mathbb{E}, A, k$)

Input: Sequence of device events $\mathbb{E} = (e_1, e_2, \ldots, e_m)$, user activity A, and integer $k \in \{1, |\mathcal{S}(A)|\}$.

Output: List $\mathbb{L}^{min,A,k}$ of all representatives of equivalence classes of $\Psi^{\min,A,k}$, in increasing order w.r.t $\prec$.

$\eta \leftarrow \left|\mathbb{S}^k(A)\right|$; $\mathbb{L}^{\min,A,k} \leftarrow$ null ; $l \leftarrow 0$; start $\leftarrow 1$; $i \leftarrow 1$;
/* finding a match of $\mathbb{S}^k(A)$ in $\mathbb{E}$(start $: m$) */
for $j := 0$ to η do $c[$ start $- 1, j] \leftarrow 0$;
while $i \leq m$ do
  $c[i, 0] \leftarrow 0$;
  for $j := 1$ to η do
    if $e_i = e_j^{A,k}$ then
      $c[i, j] \leftarrow c[i-1, j-1] + 1$
    else if $c[i-1, j] \geq c[i, j-1]$ then
      $c[i, j] \leftarrow c[i-1, j]$;
    else $c[i, j] \leftarrow c[i, j-1]$;
  if $c[i, \eta] = n$ then
    /* a new match of $\mathbb{S}^k(A)$ is found */

FIG. 4A

401B – ALGORITHM 1 (continued)

AkMatch($\mathbb{E}, A, k$)

```
12        l ← l + 1; initialize ψ_l; row ← i; col ;
13        while col > 0 do
14            if e_row = e_col^{A,k} then
15                ψ_l[col] ← row ; row ← row − 1; col ← col − 1;
16            else if c[row − 1, col] ≥ c[row, col − 1] then
17                row ← row − 1;
18            else col ← col − 1;
19        L^{min,A,k}.append(ψ_l)
20        start ← ψ_l[1] + 1; i ← start; goto 2;
21    i ← i + 1;
22 output L^{min,A,k}.
```

FIG. 4B

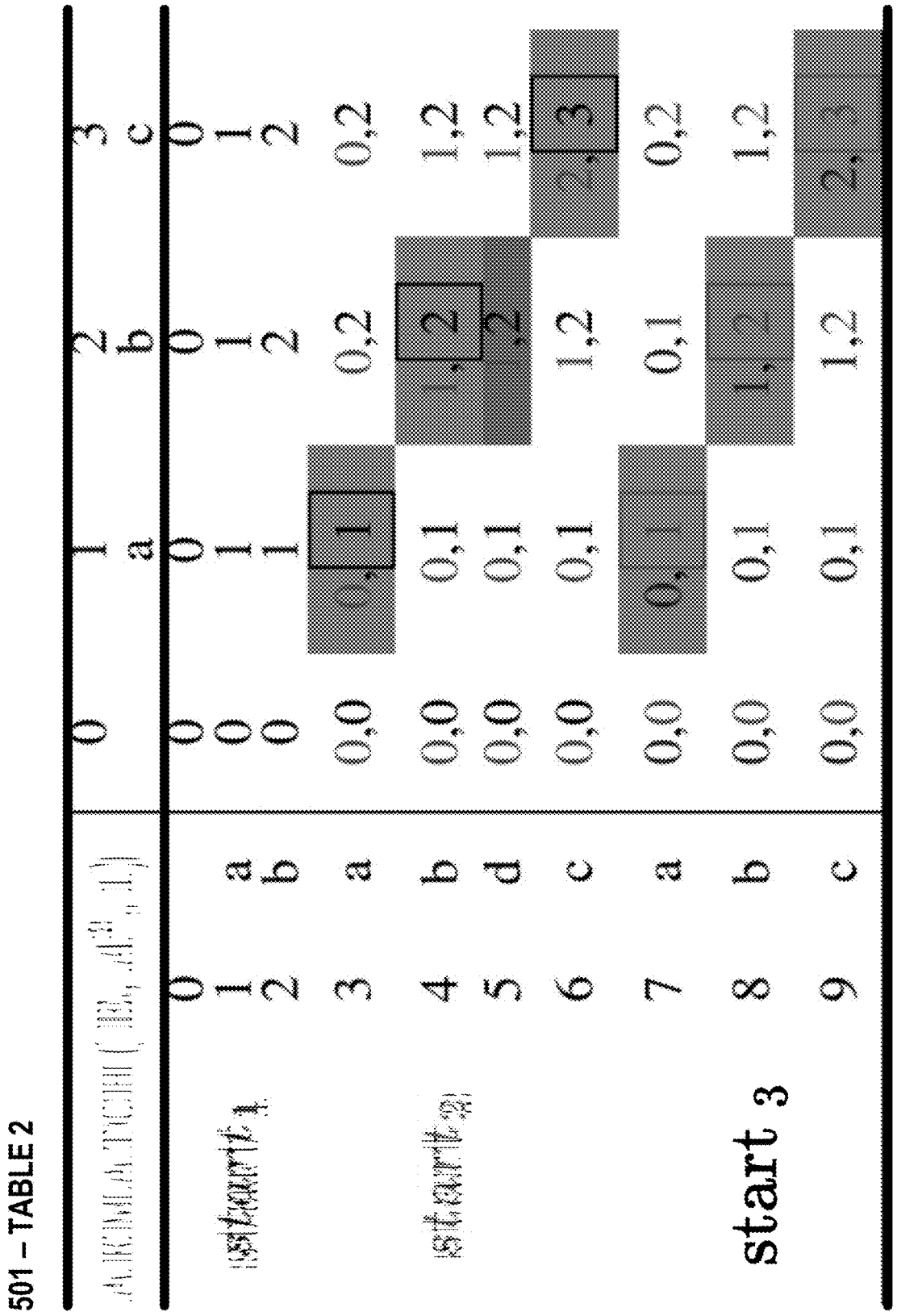


FIG. 5

601 – ALGORITHM 1 (continued)

OMatch($\mathbb{M}, w$)

Input: $\mathbb{M}$:computed match of $\mathcal{A}$ in $\mathbb{E}$; w : weight function
Output: $\mathbb{M}_w$:max weight sequence of compatible matches

$OPT[0] \leftarrow 0$
for $j = 1$ to M do
    if $(\mathbb{M}[j] \cdot w + OPT[\Re\{j\}] > OPT[j-1])$ then
        $OPT[j] \leftarrow \mathbb{M}[j] \cdot w + OPT[p[j]]$;
    else $OPT[j] \leftarrow OPT[j-1]$;
$\mathbb{M}_w \leftarrow$ null; $j \leftarrow M$;
while $j > 0$ do
    if $(\mathbb{M}[j].w + OPT[p[j]] > OPT[j-1])$ then
        $\mathbb{M}_w \cdot \text{insert}(\mathbb{M}[j]); j \leftarrow p[j]$;
    else $j \leftarrow j - 1$;
output $\mathbb{M}_w$.

FIG. 6

701A – ALGORITHM 3

SLN-1D($\mathbb{M}, w, \underline{i}, \underline{k}$)

Input: $\mathbb{M}$: computed match of $\mathcal{A}$ in $\mathbb{E}$; w : weight vector; $\underline{i}$, $\underline{k}$: index pair for variable $w(\underline{i}, \underline{k})$

Output: x_{opt} :optimal value for $w(\underline{i}, \underline{k})$; f_{opt} : loss function value with $w(\underline{i}, \underline{k})$ at optimal value

$x \leftarrow 0$; $x_{opt} \leftarrow 0$; $w(\underline{i}, \underline{k}) \leftarrow x$; $f_{opt} \leftarrow f(w)$; $\mathbb{E}_{\text{old}} \leftarrow \mathbb{E}^w$;
$OPT[0] \leftarrow 0$; $a[0] \leftarrow 0$; $\sigma\circ \leftarrow \infty$;
for $j := 1$ to M do
  if $(\mathbb{M}[j].A = A^{\underline{i}}$ and $\mathbb{M}[j].k = \underline{k})$ then
    $\delta[j] \leftarrow 1$; /* $\mathbb{M}[j].w$ is variable */
  else
    $\delta[j] \leftarrow 0$; /* $\mathbb{M}[j].w$ is constant */
  if $(\mathbb{M}[j] \cdot w + OPT[p[j]] > OPT[j-1])$ then
    $OPT[j] \leftarrow \mathbb{M}[j] \cdot w + OPT[p[j]]$
    $a[j] \leftarrow a[p[j]] + \delta[j]$;
    if $a[p[j]] + \delta[j] < a[j-1]$ then
      $\sigma \leftarrow \min\left\{\sigma, \frac{\mathbb{M}[j] \cdot w + OPT[p[j]] - OPT[j-1]}{a[j-1] - a[p[j]] - \delta[j]}\right\}$;

FIG. 7A

701B – ALGORITHM 3 (continued)

SLN-1D($\mathbb{M}, w, \underline{i}, \underline{k}$)

```
13  else if (M[j] · w + OPT[p[j]] < OPT[j − 1]) then
14      OPT[j] ← OPT[j − 1];
15      a[j] ← a[j − 1];
16      if a[p[j]] + δ[j] > a[j − 1] then
17          σ ← min{σ, (M[j] · w + OPT[p[j]] − OPT[j − 1]) / (a[j − 1] − a[p[j]] − δ[j])};
18  else
19      if a[p[j]] + δ[j] ≤ a[j − 1] then
20          OPT[j] ← OPT[j − 1];
21          a[j] ← a[j − 1];
22      else
23          OPT[j] ← M[j] · w + OPT[p[j]];
24          a[j] ← a[p[j]] + δ[j];
```

FIG. 7B

701C – ALGORITHM 3 (continued)

SLN-1D($\mathbb{M}, w, \underline{i}, \underline{k}$)

25 if $\sigma = \infty$ then $\{\sigma \leftarrow 2;$ done $\leftarrow$ true; $\}$
26 $\mu \leftarrow x + \sigma; \nu \leftarrow x + \sigma/2;$
27 $w(\underline{i}, \underline{k}) \leftarrow \nu; \mathbb{E}_{new} \leftarrow \mathbb{E}^w;$
28 if $(\mathbb{E}_{\text{new}} \neq \mathbb{E}_{\text{old}})$ or $(\nu = \sigma/2)$ then
29 $\quad f_{\text{new}} \leftarrow f(w); \mathbb{E}_{\text{old}} \leftarrow \mathbb{E}_{\text{new}};$
30 $\quad$ if $f_{\text{new}} < f_{\text{opt}}$ or $(f_{\text{new}} = f_{\text{opt}}$ and $\nu = \sigma/2)$ then
31 $\quad\quad x_{opt} \leftarrow \nu; f_{opt} \leftarrow f_{\text{new}};$
32 $w(\underline{i}, \underline{k}) \leftarrow \mu; \mathbb{E}_{new} \leftarrow \mathbb{E}^w$
33 if $(\mathbb{E}_{\text{new}} \neq \mathbb{E}_{\text{old}})$ then
34 $\quad f_{\text{new}} \leftarrow f(w); \mathbb{E}_{\text{old}} \leftarrow \mathbb{E}_{\text{new}};$
35 $\quad$ if $f_{\text{new}} < f_{\text{opt}}$ then $\{x_{\text{opt}} \leftarrow \mu; f_{\text{opt}} \leftarrow f_{\text{new}};\}$
36 if (not done) then $\{x \leftarrow \mu;$ goto 2; $\}$
37 output x_{opt}, f_{opt}.

FIG. 7C

801 – ALGORITHM 4

SLN-ND($\mathbb{M}$)

Input: $\mathbb{M}$: representative minimal matches of $\mathcal{A}$ in $\mathbb{E}$.
Output: w: optimal weight vector; $\mathbb{S}^w$: corresponding solution to E2AP.

/* initialization */
1 **for** $i := 1$ **to** n **do** {**for** $k := 1$ **to** n_i **do** $w[i,k] \leftarrow 1$;}
2 $done \leftarrow$ **false**; $f_{opt} \leftarrow f(w)$;
/* coordinate descent */
3 **while** ($done =$ **false**) **do**
/* normalize the weights */
4 $W \leftarrow 0$;
5 **for** $i := 1$ **to** n **do**
6 **for** $k := 1$ **to** n_i **do** $W \leftarrow W + w[i,k]$;
7 **for** $i := 1$ **to** n **do**
8 **for** $k := 1$ **to** n_i **do** $w[i,k] \leftarrow \frac{N \times w[i,k]}{W}$;
9 $done \leftarrow$ **true**;
10 **for** $i := 1$ **to** n **do**
/* 1D minimization */
11 **for** $k := 1$ **to** n_i **do**
12 $x_{old} \leftarrow w(i,k)$;
13 $x_{new} \leftarrow \arg\min_{w(i,k) \geq 0} f(w)$; $f_{new} \leftarrow f(w)$;
14 **if** $f_{new} < f_{opt}$ **then**
15 $done \leftarrow$ **false**; $w(i,k) \leftarrow x_{new}$; $f_{opt} \leftarrow f_{new}$;
16 **else** $w(i,k) \leftarrow x_{old}$;
17 **output** w, $\mathbb{S}^w$.

FIG. 8

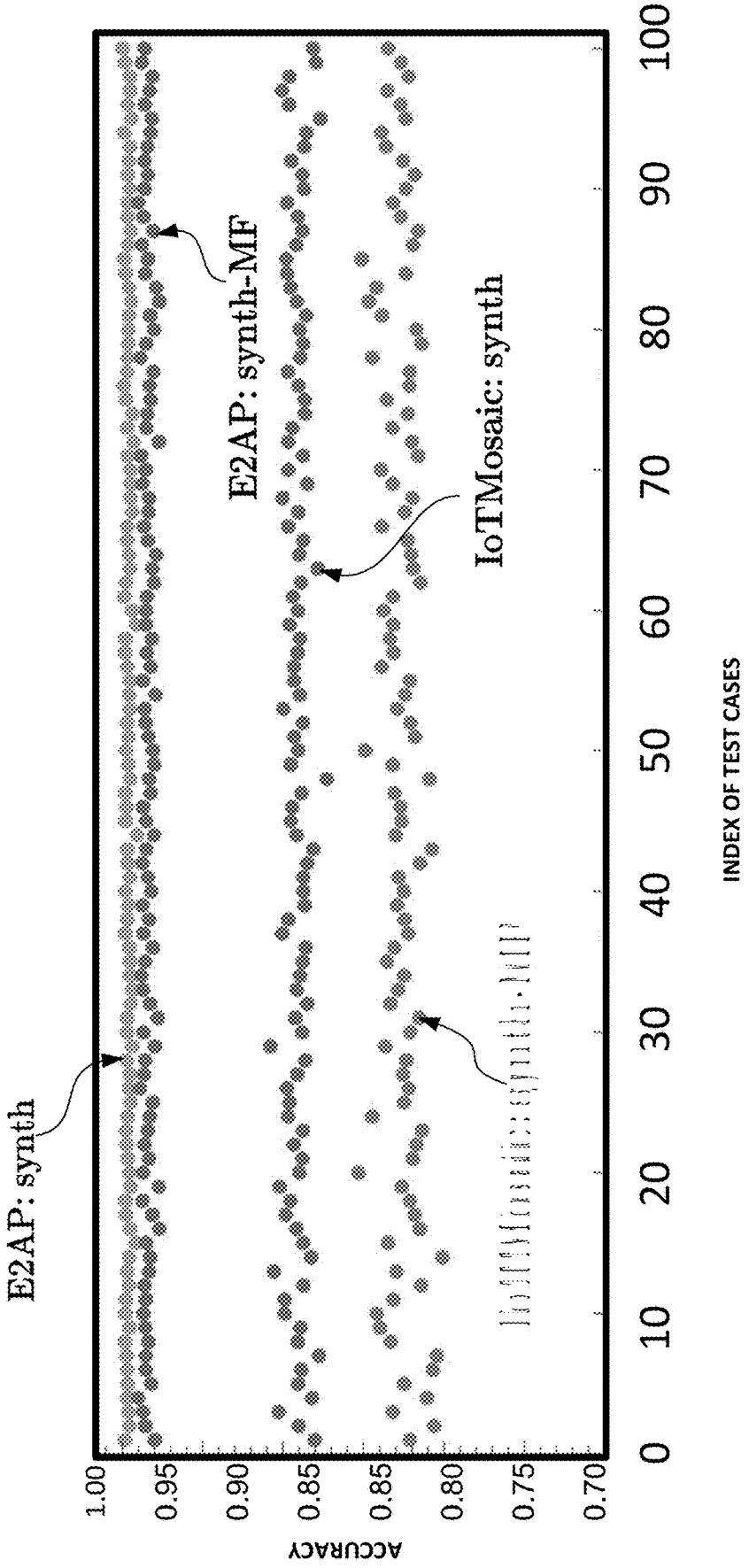

E2AP: synth
E2AP: synth-MF
IoTMosaic: synth
ACCURACY
1.00
0.95
0.90
0.85
0.85
0.80
0.75
0.70
0
10
20
30
40
50
60
70
80
90
100
INDEX OF TEST CASES

FIG. 9

1001A – TABLE 3

Data Type	#Act	#Event	IoTMosaic ($k \leq 5$) FN	FP	Accuracy
real (week)	387	2013	6	2	97.92%
real (month)	1494	7934	32	13	96.98%
real (full)	2959	15420	42	18	97.97%
synth	387	[1666, 1941]	38.76	0.44	88.77%
synth	1494	[6708, 7202]	147.50	0.92	88.99%
synth	2959	[13382, 14264]	290.08	1.36	89.09%
synth	10000	[46122, 47496]	984.61	4.67	89.04%
synth	30000	[139427, 141632]	2936.63	12.31	89.11%
synth-MF	387	[1337, 1590]	49.45	19.49	81.69%
synth-MF	1494	[5645, 6009]	183.28	68.78	82.60%
synth-MF	2959	[11161, 11855]	361.27	137.15	82.64%
synth-MF	10000	[38312, 39401]	1221.87	460.44	82.65%
synth-MF	30000	[115946, 117785]	3648.52	1385.25	82.71%

FIG. 10A

1001B – TABLE 3 (continued)

Data Type	#Act	#Event	E2AP FN	E2AP FP	E2AP Accuracy	E2AP $d(A)$	E2AP $d_N(A)$	E2AP $d(E)$	E2AP $d_N(E)$
real (week)	387	2013	2	2	98.97%	4	0.0103	22	0.0109
real (month)	1494	7934	2	12	99.06%	14	0.0094	104	0.0131
real (full)	2959	15420	2	16	99.39%	18	0.0061	142	0.0092
synth	387	[1666, 1941]	1.50	0.82	99.61%	1.52	0.0039	2.62	0.0014
synth	1494	[6708, 7202]	5.23	2.58	99.60%	5.96	0.0040	10.58	0.0015
synth	2959	[13382, 14264]	9.27	4.58	99.60%	11.05	0.0037	20.33	0.0015
synth	10000	[46122, 47496]	29.77	12.41	99.60%	40.46	0.0040	73.21	0.0016
synth	30000	[139427, 141632]	82.91	33.19	99.61%	117.04	0.0039	212.84	0.0015
synth-MF	387	[1337, 1590]	3.90	3.38	98.33%	6.47	0.0167	1.99	0.0013
synth-MF	1494	[5645, 6009]	10.89	8.44	98.32%	25.02	0.0167	8.50	0.0015
synth-MF	2959	[11161, 11855]	19.10	14.62	98.05%	47.07	0.0159	15.67	0.0014
synth-MF	10000	[38312, 39401]	54.78	39.67	98.44%	155.74	0.0156	53.31	0.0014
synth-MF	30000	[115946, 117785]	156.95	111.75	98.45%	465.13	0.0155	160.43	0.0014

FIG. 10B

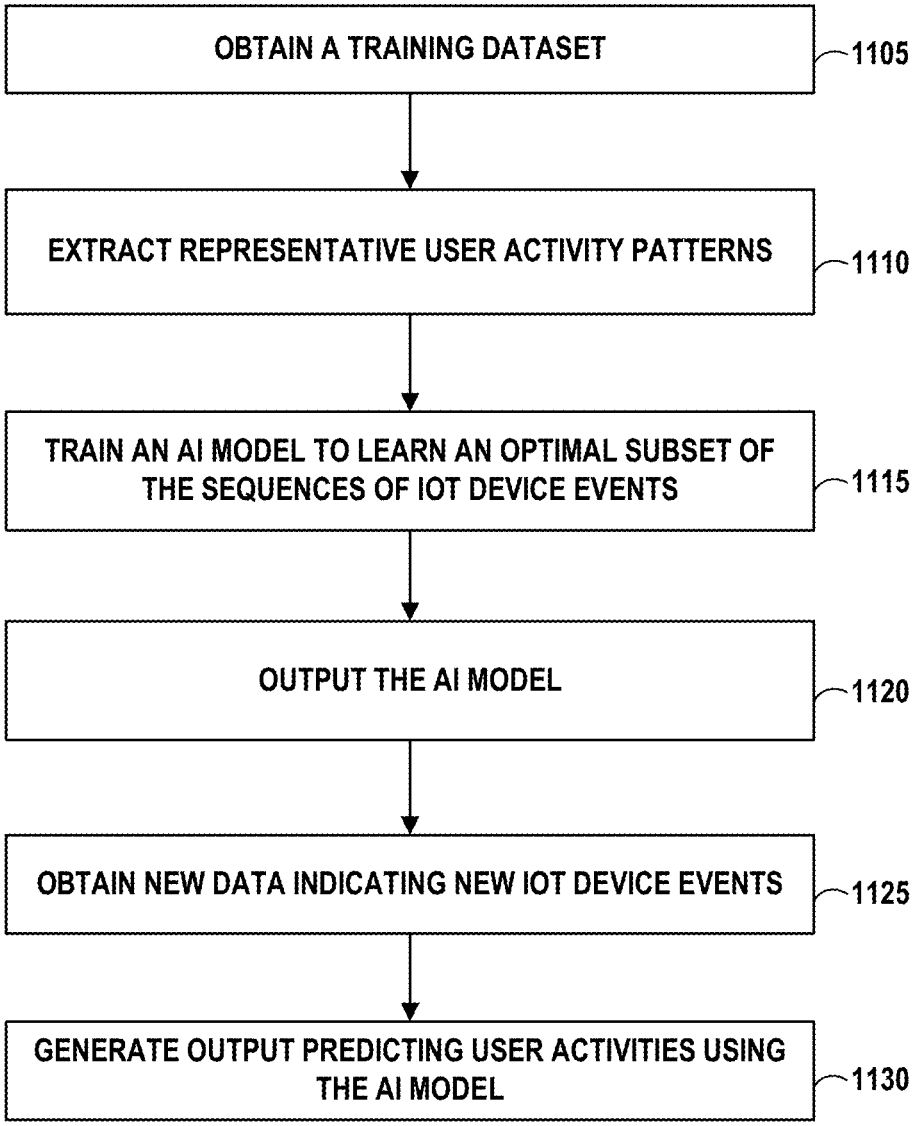

OBTAIN A TRAINING DATASET
1105
EXTRACT REPRESENTATIVE USER ACTIVITY PATTERNS
1110
TRAIN AN AI MODEL TO LEARN AN OPTIMAL SUBSET OF THE SEQUENCES OF IOT DEVICE EVENTS
1115
OUTPUT THE AI MODEL
1120
OBTAIN NEW DATA INDICATING NEW IOT DEVICE EVENTS
1125
GENERATE OUTPUT PREDICTING USER ACTIVITIES USING THE AI MODEL
1130

FIG. 11

INFERRING USER ACTIVITIES FROM INTERNET OF THINGS (IOT) CONNECTED DEVICE EVENTS USING MACHINE LEARNING BASED ALGORITHMS

CLAIM OF PRIORITY

[0001] This application claims the benefit of U.S. Patent Application No. 63/506,316, filed Jun. 5, 2023, the entire contents of which is incorporated herein by reference.

GOVERNMENT RIGHTS AND GOVERNMENT AGENCY SUPPORT NOTICE

[0002] This invention was made with government support under 1717197, 1816995 and 2007469 awarded by the National Science Foundation. The government has certain rights in the invention.

TECHNICAL FIELD

[0003] This disclosure generally relates to the field of artificial intelligence and machine learning via computational systems and more particularly, to systems, methods, and apparatuses for inferring user activities from Internet of Things (IoT) connected device events using machine learning based algorithms.

BACKGROUND

[0004] The subject matter discussed in the background section should not be assumed to be prior art merely as a result of its mention in the background section. Similarly, a problem mentioned in the background section or associated with the subject matter of the background section should not be assumed to have been previously recognized in the prior art. The subject matter in the background section merely represents different approaches, which in and of themselves may also correspond to embodiments of the claimed inventions.

[0005] Machine learning models have various applications to automatically process inputs and produce outputs considering situational factors and learned information to improve output quality. One area where machine learning models, and neural networks in particular, provide high utility is in the field of image processing.

[0006] Within the context of machine learning and with regard to deep learning specifically, a Convolutional Neural Network (CNN, or ConvNet) is a class of deep neural networks, very often applied to analyzing visual imagery. Convolutional Neural Networks are regularized versions of multilayer perceptrons. Multilayer perceptrons are fully connected networks, such that each neuron in one layer is connected to all neurons in the next layer, a characteristic which often leads to a problem of overfitting of the data and the need for model regularization. Convolutional Neural Networks also seek to apply model regularization, but with a distinct approach. Specifically, CNNs take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Consequently, on the scale of connectedness and complexity, CNNs are on the lower extreme.

SUMMARY

[0007] In general, this disclosure is directed to improved techniques for inferring user activities from Internet of Things (IoT) connected device events using machine learning based algorithms.

[0008] The Internet of Things or "IT" describes physical network connected objects or groups of such objects having sensors, processing capabilities, software, and other computational technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Commonly, but not necessarily, connected with a public Internet, such IoT devices are individually addressable and very often operate autonomously, without requiring user input once they are provisioned and connected with a network.

[0009] The rapid and ubiquitous deployment of Internet of Things (IoT) in smart homes has created unprecedented opportunities to automatically extract environmental knowledge, awareness, and intelligence. Attempts have been made to utilize either machine learning approaches or deterministic approaches to infer IoT device events and/or user activities from network traffic in smart homes.

[0010] Described herein are solutions and improved methodologies via which to overcome the difficulties and problems of inferring user activity patterns from a sequence of device events.

[0011] Unfortunately, prior known techniques fail to adequately capture, analyze, and provide meaningful predictive output on the basis of available IoT data.

[0012] What is needed is an improved technique for using unsupervised learning algorithms which are capable of making inferences that are more adaptive to varying scenarios, such as IoT device malfunctions versus legitimate inferences from user activity.

[0013] The present state of the art may therefore benefit from the systems, methods, and apparatuses for inferring user activities from Internet of Things (IoT) connected device events using machine learning based algorithms, as is described herein.

[0014] In at least one example, one or more processors of a computing device are configured to perform a computer-implemented method. Such a method may include processing circuitry executing an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model. According to such an example, processing circuitry may obtain, using the IoT learning framework, a training dataset indicating at least sequences of IoT device events and extract representative user activity patterns from the sequences of IoT device events indicated by the training dataset. In such an example, processing circuitry may train, using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold. The processing circuitry may output, using the IoT learning framework, the AI model and obtain new data indicating new sequences of IoT device events not indicated by the training dataset. In such an example, processing circuitry may generate, using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

[0015] In at least one example, a system includes processing circuitry; non-transitory computer readable media; and

instructions that, when executed by the processing circuitry, configure the processing circuitry to perform operations. In such an example, processing circuitry may configure the system to execute an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model. According to such an example, processing circuitry may obtain, using the IoT learning framework, a training dataset indicating at least sequences of IoT device events and extract representative user activity patterns from the sequences of IoT device events indicated by the training dataset. In such an example, processing circuitry may train, using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold. The processing circuitry may output, using the IoT learning framework, the AI model and obtain new data indicating new sequences of IoT device events not indicated by the training dataset. In such an example, processing circuitry may generate, using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

[0016] In one example, there is computer-readable storage media having instructions that, when executed, configure processing circuitry to perform operations. Such operations may include executing an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model. According to such an example, processing circuitry may obtain, using the IoT learning framework, a training dataset indicating at least sequences of IoT device events and extract representative user activity patterns from the sequences of IoT device events indicated by the training dataset. In such an example, processing circuitry may train, using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold. The processing circuitry may output, using the IT learning framework, the AI model and obtain new data indicating new sequences of IoT device events not indicated by the training dataset. In such an example, processing circuitry may generate, using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

[0017] The details of one or more examples of the disclosure are set forth in the accompanying drawings and the description below. Other features, objects, and advantages will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF DRAWINGS

[0018] FIG. **1** is a block diagram illustrating further details of one example of computing device, in accordance with aspects of the disclosure.

[0019] FIG. **2** depicts an example architecture of the Internet of Things (IoT), each being connected with a network, such as a public Internet, in accordance with aspects of the disclosure.

[0020] FIGS. **3**A, **3**B, and **3**C set forth Table 1 providing a listing of commonly used notations used in accordance with aspects of the disclosure.

[0021] FIGS. **4**A and **4**B set forth Algorithm 1 depicting an example implementation of the AkMatch functionality for computing the minimal and the lightest amongst the available matches, in accordance with aspects of the disclosure.

[0022] FIG. **5** sets forth Table 2 depicting an example graphical illustration of the AkMatch functionality, in accordance with aspects of the disclosure.

[0023] FIG. **6** sets forth Algorithm 2 depicting an example implementation of the OMatch ($\mathbb{M}$,w) functionality for computing a compatible sub-sequence of user activity patterns, in accordance with aspects of the disclosure.

[0024] FIGS. **7**A, **7**B, and **7**C set forth Algorithm 3 depicting an example implementation of the SLN-1D functionality for computing an optimal solution to the 1-D problem, in accordance with aspects of the disclosure.

[0025] FIG. **8** sets forth Algorithm 4 depicting an example implementation of the SLN-ND functionality for applying a round-robin coordinate descent approach as part of solving an optimization problem, in accordance with aspects of the disclosure.

[0026] FIG. **9** depicts a chart of the various performances of E2AP and IoTMosaic on synth/synth-MF test cases, in accordance with aspects of the disclosure.

[0027] FIGS. **10**A and **10**B set forth Table 3A depicting results on real data, with one case per row and further depicting synthetic data as an average over 100 cases per row, in accordance with aspects of the disclosure.

[0028] FIG. **11** is a flow chart illustrating an example mode of operation for the computing device to infer user activities from Internet of Things (IoT) connected device events using machine learning based algorithms, in accordance with aspects of the disclosure.

[0029] Like reference characters denote like elements throughout the text and figures.

DETAILED DESCRIPTION

[0030] Aspects of the disclosure provide improved techniques for inferring user activities from Internet of Things (IoT) connected device events using machine learning based algorithms.

[0031] The Internet of Things or "IoT" describes physical network connected objects or groups of such objects having sensors, processing capabilities, software, and other computational technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Commonly, but not necessarily, connected with a public Internet, such IoT devices are individually addressable and very often operate autonomously, without requiring user input once they are provisioned and connected with a network.

[0032] The rapid and ubiquitous deployment of Internet of Things (IoT) in smart homes has created unprecedented opportunities to automatically extract environmental knowledge, awareness, and intelligence. Attempts have been made to utilize either machine learning approaches or deterministic approaches to infer IoT device events and/or user activities from network traffic in smart homes.

[0033] Described herein are solutions and improved methodologies via which to overcome the difficulties and problems of inferring user activity patterns from a sequence of device events. For instance, example techniques set forth herein operate by first deterministically extracting a small number of representative user activity patterns from the sequence of device events, then applying unsupervised

learning to compute an optimal subset of these user activity patterns to infer user activities. Based on extensive experiments with sequences of device events triggered by 2,959 real user activities and up to 30,000 synthetic user activities, experimental results demonstrate that the disclosed scheme is resilient to device malfunctions and transient failures/delays, and outperforms all prior known state-of-the-art solutions.

[0034] FIG. **1** is a block diagram illustrating further details of one example of computing device, in accordance with aspects of the disclosure. FIG. **1** illustrates only one particular example of computing device **100**. Many other examples of computing device **100** may be used in other instances.

[0035] As shown in the specific example of FIG. **1**, computing device **100** may include processing circuitry **199** including one or more processors **105** and memory **104**. Computing device **100** may further include network interface **106**, one or more storage devices **108**, user interface **110**, and power source **112**. Computing device **100** may also include an operating system **114**. Computing device **100**, in one example, may further include one or more applications **116**, such as predicted user activities **163** and unsupervised learning **184**. One or more other applications **116** may also be executable by computing device **100**. Components of computing device **100** may be interconnected (physically, communicatively, and/or operatively) for inter-component communications.

[0036] Operating system **114** may execute various functions including executing trained AI model **193** and performing AI model training. As shown here, operating system **114** executes an Internet-of-Things (IoT) learning framework **165** (IoT learning framework **165** hereinafter) which includes both IoT event sequences **161** and threshold weights **162** by which to configure and adjust learning and predictive characteristics of AI model **193** trained via IoT learning framework **165**. Threshold weights **162** may receive as input, a training dataset **139** upon which network weights utilized by IoT learning framework **165** may be adjusted or generated. IoT learning framework **165** further includes user activity patterns **167** which may be derived or predicted by IoT learning framework **165** utilizing learnings derived from training dataset **139**.

[0037] Computing device **100** may perform techniques for inferring user activities from Internet of Things (IoT) connected device events using machine learning based algorithms, including performing AI model **193** training using training dataset **139** including, for example, learning techniques for solving the E2AP and E2A problems and determining weights for the patterns using unsupervised learning. IoT learning framework **165** may train and generate as output, trained AI model **193**. Computing device **100** may provide trained AI model **193** as output to a connected user device via user interface **110**.

[0038] In some examples, processing circuitry including one or more processors **105**, implements functionality and/or process instructions for execution within computing device **100**. For example, one or more processors **105** may be capable of processing instructions stored in memory **104** and/or instructions stored on one or more storage devices **108**.

[0039] Memory **104**, in one example, may store information within computing device **100** during operation. Memory **104**, in some examples, may represent a computer-readable storage medium. In some examples, memory **104** may be a temporary memory, meaning that a primary purpose of memory **104** may not be long-term storage. Memory **104**, in some examples, may be described as a volatile memory, meaning that memory **104** may not maintain stored contents when computing device **100** is turned off. Examples of volatile memories may include random access memories (RAM), dynamic random-access memories (DRAM), static random-access memories (SRAM), and other forms of volatile memories. In some examples, memory **104** may be used to store program instructions for execution by one or more processors **105**. Memory **104**, in one example, may be used by software or applications running on computing device **100** (e.g., one or more applications **116**) to temporarily store data and/or instructions during program execution.

[0040] One or more storage devices **108**, in some examples, may also include one or more computer-readable storage media. One or more storage devices **108** may be configured to store larger amounts of information than memory **104**. One or more storage devices **108** may further be configured for long-term storage of information. In some examples, one or more storage devices **108** may include non-volatile storage elements. Examples of such non-volatile storage elements may include magnetic hard disks, optical discs, floppy disks, Flash memories, or forms of electrically programmable memories (EPROM) or electrically erasable and programmable (EEPROM) memories.

[0041] Computing device **100**, in some examples, may also include a network interface **106**. Computing device **100**, in such examples, may use network interface **106** to communicate with external devices via one or more networks, such as one or more wired or wireless networks. Network interface **106** may be a network interface card, such as an Ethernet card, an optical transceiver, a radio frequency transceiver, a cellular transceiver or cellular radio, or any other type of device that may send and receive information. Other examples of such network interfaces may include BLUETOOTH®, 3G, 4G, 1G, LTE, and WI-FI® radios in mobile computing devices as well as USB. In some examples, computing device **100** may use network interface **106** to wirelessly communicate with an external device such as a server, mobile phone, or other networked computing device.

[0042] User interface **110** may include one or more input devices **111**, such as a touch-sensitive display. Input device **111**, in some examples, may be configured to receive input from a user through tactile, electromagnetic, audio, and/or video feedback. Examples of input device **111** may include a touch-sensitive display, mouse, keyboard, voice responsive system, video camera, microphone or any other type of device for detecting gestures by a user. In some examples, a touch-sensitive display may include a presence-sensitive screen.

[0043] User interface **110** may also include one or more output devices, such as a display screen of a computing device or a touch-sensitive display, including a touch-sensitive display of a mobile computing device. One or more output devices, in some examples, may be configured to provide output to a user using tactile, audio, or video stimuli. One or more output devices, in one example, may include a display, sound card, a video graphics adapter card, or any other type of device for converting a signal into an appropriate form understandable to humans or machines. Additional examples of one or more output devices may include

a speaker, a cathode ray tube (CRT) monitor, a liquid crystal display (LCD), or any other type of device that may generate intelligible output to a user.

[0044] Computing device **100**, in some examples, may include power source **112**, which may be rechargeable and provide power to computing device **100**. Power source **112**, in some examples, may be a battery made from nickel-cadmium, lithium-ion, or other suitable material.

[0045] Examples of computing device **100** may include operating system **114**. Operating system **114** may be stored in one or more storage devices **108** and may control the operation of components of computing device **100**. For example, operating system **114** may facilitate the interaction of one or more applications **116** with hardware components of computing device **100**.

[0046] FIG. **2** depicts an example architecture of the Internet of Things (IoT) **200**, each being connected with a network, such as a public Internet, in accordance with aspects of the disclosure.

[0047] The ubiquitous and heterogeneous deployment of Internet of Things (IoT) devices in smart homes has created new opportunities to extract knowledge, awareness, and intelligence via monitoring and understanding interactions of the devices with their environments and users. A number of studies focused on discovering meaningful information from network traffic of IoT devices in smart homes, even though much of the traffic is often encrypted over secure wireless networks or via IoT application-level encryption.

[0048] For example, prior techniques have attempted to use machine learning (ML) models to infer device events from network packets, albeit with limited success. Example models PingPong and IoTAthena models utilized deterministic algorithms for device event extraction based on the observation that every device event generates a repeatable sequence of network packets. Other studies have explored the feasibility of inferring user activities with IoT devices in smart homes. For example, one such technique utilized an unsupervised learning method to discover user activities based on information collected by sensors deployed in a smart home. One study demonstrated the possibility of passively sniffing wireless-only network traffic of IoT devices for detecting and identifying IoT device types and user activities in smart homes from an adversary perspective. Still further, User Activity Inference (UAI) and an approximate matching-based algorithm has been explored to infer a multi-set of user activities from the network traffic of IoT devices, which was actively collected at programmable home routers from a trusted home user perspective.

[0049] However, reconstructing real-world user activities in smart homes and matching them with the ground-truth requires an exact sequence of user activities, rather than a multi-set of user activities produced by IoTMosaic.

[0050] Towards this end, the problems of Events to Activities (E2A) and Events to Activity Patterns (E2AP) were studied from a trusted home user perspective for inferring a sequence of user activities from a sequence of IoT device events, which can be extracted from network traffic using existing methods such as PingPong and IoTAthena. A two-phase scheme was specially designed and configured employing both deterministic algorithms and machine learning techniques for solving the E2AP and E2A problems.

[0051] In Phase 1, a small number of representative matches of the activity patterns is computed. It is proven that the solution computed based on these representative matches is as good as any solution that can be obtained by considering all possible matches. In Phase 2, an efficient algorithm is described for computing a compatible set of matches of user activity patterns with maximum total weight, for any given weight assignment to the matches.

[0052] An unsupervised learning algorithm was also designed and configured for computing a good set of weights. While the loss function for the problem is non-differentiable, an exact algorithm was specifically designed for minimizing the loss function over any one of the weight variables. Similar to many ML algorithms that build optimization models and learn optimal parameters, the implementation described herein utilizes a coordinate descent algorithm designed for solving E2AP which converges in a finite number of iterations.

[0053] Therefore, practice of the described techniques provides at least the following benefits: Firstly, the problem of inferring a sequence of user activities and their patterns in smart homes is formulated from a sequence of device events extracted from network traffic. Secondly, it is proven that one can concentrate on a small number of representative matches of user activity patterns and design an efficient algorithm for computing these representatives. Thirdly, an efficient algorithm described herein was designed and implemented to compute a compatible set of matches of user activity patterns with maximum total weight for any given weight assignment. A complementary algorithm was then designed and implemented to compute an optimal set of weights in a finite number of iterations via unsupervised learning. Fourthly, extensive experiments were conducted with both real and synthetic data and demonstrate that the algorithm is robust and outperforms the state-of-the-art solution.

Problem Formulation and Basics:

[0054] Research suggests that a user activity in a smart home may be inferred from the sequence of IoT device events triggered by the activity. For example, one of the user activities studied in attempts indicates that: "[a] person without [a] key entering the home from the front door during the day" usually triggers the sequence of 10 IoT device events, including: Ring doorbell motion detection, Ring spotlight motion detection, Ring doorbell ringing, Ring doorbell stream on, Ring doorbell stream off, August lock manual unlocking, Tessan contact sensor open, Tessan contact sensor close, August lock manual locking, Arlo Q camera motion detection, collectively involving 5 devices.

[0055] Intuitively, observing this sequence of 10 device events in a very short period of time suggests that the above user activity very likely has happened. If a device malfunctions when a user activity occurs, the device events corresponding to this device will be missing from the sequence. For example, if the Tessan contact sensor malfunctions, the events Tessan contact sensor open and Tessan contact sensor close will be missing in the corresponding sequence of device events. If the Ring doorbell is in sleeping mode when the user activity occurs, the event Ring doorbell motion detection will be delayed. Because the amount of delay varies significantly in the experiments, device events were thus considered to be missing as well.

[0056] The experiments use $\mathcal{A}=\{A^1, A^2, \ldots, A^{|\mathcal{A}|}\}$ of user activities. Each occurrence of a user activity $A\in\mathcal{A}$ triggers a sequence of IoT device events

$$\mathbb{S}^1(A) = \left(e_1^{A,1}, e_2^{A,1}, \ldots, e_{|\mathbb{S}^1(A)|}^{A,1}\right)$$

or a subsequence of $\mathbb{S}^1$ (A), where the missing events correspond to devices that malfunction when A occurs. The experiments set use $|\mathcal{A}|$ to denote the cardinality of set A and use $|\mathbb{S}|$ to denote the length of sequence $\mathbb{S}$. The experiments set use $\mathbb{A}=(A_1, A_2, \ldots, A_{|\mathbb{A}|})$ to denote a sequence of user activities. The examples provided use superscripts in the description of (distinct) user activities in set $\mathcal{A}$, and use subscripts in the description of (not necessarily distinct) user activities in sequence $\mathbb{A}$. The terms A^1 and A^2 denote two distinct elements in set A. The terms A_1 and A_2 denote the first and second elements in sequence $\mathbb{A}$, respectively.

[0057] For a user activity A, the examples use $\mathcal{S}(A)=\{\mathbb{S}^1(A), \mathbb{S}^2(A), \ldots, \mathbb{S}^{|\mathcal{S}(A)|}(A)\}$ to denote the set of distinct sequences of device events that could be triggered by A, where

$$\mathbb{S}^k(A) = \left(e_1^{A,k}, e_2^{A,k}, \ldots, e_{|\mathbb{S}^k(A)|}^{A,k}\right)$$

for k=1, 2, . . . , |S(A)|, and $\mathbb{S}^k$ (A) is a subsequence of $\mathbb{S}^1$ (A) for k>1. The examples call $\mathcal{S}$ (A) the set of possible patterns of A. In the experiments, $\mathcal{S}$ (A) is extracted through observing the device event sequences triggered by repeated occurrences of A.

[0058] Each IoT device may correspond to multiple events. Each user may correspond to multiple user activities, and two different user activities may both involve a common device event. IoT learning framework **165** may focus AI model training on user activities (rather than users) and device events (rather than devices). Several examples are used to illustrate important concepts and algorithms. For ease of understanding, the following settings were utilized for all examples.

[0059] Example Setting: $\mathbb{E}=(a, b, a, b, d, c, a, b, c)$ is the sequence of device events. $\mathcal{A}=\{A^1, A^2\}$ is the set of user activities. The set of possible patterns for A^1 (A^2, respectively) is $\mathcal{S}(A^1)=\{(a, b)\}$($\mathcal{S}(A^2)=\{(a, b, c), (a, b)\}$.

[0060] The sequence of device events triggered by a sequence $\mathbb{A}$ of user activities, denoted by E($\mathbb{A}$), satisfies Equation 1, set forth below, as follows:

$$E(\mathbb{A}) \in \{Z_1 \| Z_2 \| \ldots \| Z_{|\mathbb{A}|} | Z_j \in \mathcal{S}(A_j), \forall\, j = 1, \ldots, |\mathbb{A}|\}.$$

[0061] Example 1: Let $\mathbb{A}^1=(A^1, A^2, A^2)$ and $\mathbb{A}^2=(A^2, A^2, A^2)$ be two sequences of user activities, where A^1 and A^2 are as in the example setting. Then E($\mathbb{A}^1$) could be any of (a, b, a, b, c, a, b, c), (a, b, a, b, c, a, b), (a, b, a, b, a, b, c), or (a, b, a, b, a, b). Similarly, E($\mathbb{A}^2$) could be any of (a, b, c, a, b, c, a, b, c), (a, b, c, a, b, c, a, b), (a, b, c, a, b, a, b, c), (a, b, c, a, b, a, b), (a, b, a, b, c, a, b, c), (a, b, a, b, c, a, b), (a, b, a, b, a, b, c), or (a, b, a, b, a, b).

[0062] This example shows that the same sequence of user activities may trigger different sequences of device events, while different sequences of user activities may trigger the same sequence of device events. The nondeterministic mapping E maps a sequence of user activities to a sequence of device events. Also of interest is the reverse problem, specifically: inferring the sequence of user activities from a given sequence of IoT device events. This is called the E2A problem (Events to Activities), defined in the following.

[0063] Problem 1 (E2A): Let $\mathcal{A}=\{A^1, A^2, \ldots, A^{|\mathcal{A}|}\}$ be a set of user activities. For each A$\in\mathcal{A}$, its set of possible patterns $\mathcal{S}$ (A) is known. Given a sequence of IoT device events $\mathbb{E}=(e_1, e_2, \ldots, e_m)$, the E2A problem seeks for a sequence $\mathbb{A}=(A^1, A^2, \ldots, A_{|\mathbb{A}|})$ of user activities in A that is most likely to trigger the sequence of device events $\mathbb{E}$.

[0064] Further described herein is the Events to Activity Patterns problem (E2AP) to infer a sequence of user activities together with their patterns. The solution to this problem can be used to provide accurate quantification of the solution quality.

[0065] Problem 2 (E2AP): Let $\mathcal{A}=\{A^1, A^2, \ldots, A^{|\mathcal{A}|}\}$ be a set of user activities. For each A$\in\mathcal{A}$ A$\in$A, its set of possible patterns $\mathcal{S}$ (A) is known. Given a sequence of IoT device events $\mathbb{E}=(e_1, e_2, \ldots, e_m)$, the E2AP problem seeks for a sequence $\mathbb{A}=(A_1, A_2, \ldots, A_{|\mathbb{A}|})$ of user activities in A together with pattern $\mathbb{S}_j^{k_j} \in \mathcal{S}(A_j)$ for j=1, 2, . . . , |$\mathbb{A}$|, such that $\mathbb{S}_1^{k_1}\|\mathbb{S}_2^{k_2}\|\ldots\|\mathbb{S}_{|\mathbb{A}|}^{k_{|\mathbb{A}|}}$ is as close to as possible.

[0066] A solution to the E2AP problem consists of a sequence $\mathbb{A}=(A_1, A_2, \ldots, A_{|\mathbb{A}|})$ of user activities and a corresponding sequence $\mathbb{S}=(\mathbb{S}_1^{k_1}, \mathbb{S}_2^{k_2}, \ldots, \mathbb{S}_{|\mathbb{A}|}^{k_{|\mathbb{A}|}})$ of activity patterns. One can use $\mathbb{A}$ as a solution to the E2A problem. The edit distance between $\mathbb{E}$ and $\mathbb{S}_1^{k_1}|\mathbb{S}_2^{k_2}\|\ldots|\mathbb{S}_{|\mathbb{A}|}^{k_{|\mathbb{A}|}}$ can be used as a metric to quantify the solution quality.

[0067] The E2AP problem is closely related to the UAI problem. With the goal of inferring user activities in a smart home being a known problem, the specific goals and techniques used by the methodologies described herein are different from any prior known technique.

[0068] Prior solutions to UAI produce a multi-set of user activities; whereas E2AP produces a sequence of user activity patterns. Note that one may use a sequence of user activity patterns to produce a sequence of user activities which in turn can be used to produce a multi-set of user activities, but not vice versa. A sequence of user activity patterns can be directly compared with the sequence of device events to measure the quality of a solution for E2AP. Such comparison is not possible if the output is a multi-set or a sequence of user activities (without the patterns).

[0069] In UAI solutions, each user activity A$\in\mathcal{A}$ has one signature, which is the full sequence of device events that may be triggered by A (corresponding to $\mathbb{S}^1$ (A) as used herein), but note that it may trigger a subsequence of the signature. When computing the matching of a signature, prior attempts relied on k-approximate subsequence matching, and gave high priorities to the full sequence of the signature over a partial sequence. In E2AP, all possible signature patterns are considered independently, and the described implementation decides the weights for the patterns using unsupervised learning, where the aim is to minimize the edit distance between the given sequence of device events and the concatenation of the sequence of computed activity patterns. This makes the ML-based

scheme resilient to device malfunctions and transient failures/delays, a significant advantage over prior known algorithms.

[0070] A two-phase scheme for solving E2AP was therefore designed and implemented.

[0071] Phase 1 computes a small number of representative matches in E for each possible pattern $\mathbb{S}^k(A^i)$ of each user activity $A_i \in \mathcal{A}$. It is proven that the solution computed based on the small number of representative matches is as good as any solution that can be obtained by considering all matches. Phase 2 consists of Phase 2A and Phase 2B. Phase 2A computes a compatible sequence $\mathbb{S}^w$ of representative matches of user activity patterns with maximum total weight, for any given weight assignment w of the matches. The edit distance between $\mathbb{E}$ and the concatenation of the user activity patterns in $\mathbb{S}^w$ is the value of the loss function $f(w)$. Phase 2B aims to compute an optimal weight w via unsupervised learning. Both Phase 1 and Phase 2B are executed once. Phase 2A is executed multiple times, one for each evaluation of $f(w)$ within Phase 2B.

[0072] FIGS. **3**A, **3**B, and **3**C set forth Table 1 at elements **301**A, **301**B, and **301**C, respectively, providing a listing of commonly used notations used in accordance with aspects of the disclosure.

[0073] FIGS. **4**A and **4**B set forth Algorithm 1 at element **401**A and **401**B, respectively, depicting an example implementation of the AkMatch functionality for computing the minimal and the lightest amongst the available matches, in accordance with aspects of the disclosure.

[0074] Matching of a possible activity pattern in the sequence of device events provides a powerful capability as described below.

[0075] Definition 1 (match and minimal match): Let $\mathbb{E}=(e_1, e_2, \ldots, e_m)$ be the sequence of device events. Let $A \in \mathcal{A}$ be a user activity with $\mathcal{S}(A)=\{\mathbb{S}^1(A), \mathbb{S}^2(A), \ldots, \mathbb{S}^{|\mathcal{S}(A)|}(A)\}$ as its set of possible patterns, where for

$$\mathbb{S}^k(A) = \left(e_1^{A,k}, e_2^{A,k}, \ldots, e_{|\mathbb{S}^k(A)|}^{A,k}\right)$$

$k=1, 2, \ldots, |\mathcal{S}(A)|$.

[0076] A match of $\mathbb{S}^k(A)$ in $\mathbb{E}$, denoted by ψ_i^{Ak}, $i \in \mathbb{Z}^+$, is a sequence of positive integers

$$\left(\psi_{i,1}^{A,k}, \psi_{i,2}^{A,k}, \ldots, \psi_{i,|\mathbb{S}^k(A)|}^{A,k}\right),$$

according to Equation 2, set forth below as follows:

$$1 \le \psi_{i,j}^{A,k} < \psi_{i,j'}^{A,k} \le m, \forall\, 1 \le j < j' \le |\mathbb{S}^k(A)|,$$

and further according to Equation 3, set forth below, as follows:

$$e_{\psi_{i,k}^{A,k}} = e_j^{A,k}, \forall\, 1 \le j \le |\mathbb{S}^k(A)|.$$

[0077] The term

$$\left[\psi_{i,1}^{A,k}, \psi_{i,|\mathbb{S}^k(A)|}^{A,k}\right]$$

represents the interval of $\psi_i^{A,k}$, denoted as $\psi_i^{A,k}$.interval. The term $\Psi^{A,k}$ is used to denote the set of all matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$. A match $\psi_i^{A,k}$ of $\mathbb{S}^k(A)$ in $\mathbb{E}$ is called minimal, if there is no match $\psi_{i'}^{A,k}$ of $\mathbb{S}^k(A)$ in $\mathbb{E}$ whose interval is a proper subset of the interval of $\psi_i^{A,k}$. The term $\Psi^{min,A,k}$ denotes the set of all minimal matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$.

[0078] Example 2: Set A to A^2 in the example setting. Then $\mathbb{S}^1(A)=(a, b, c)$ has 9 matches in $\mathbb{E}$: $\psi_1^{A,1}=(1,2,6)$ $\psi_2^{A,1}=(1,2,9)$, $\psi_3^{A,1}=(1,4,6)$, $\psi_4^{A,1}=(1,4,9)$, $\psi_5^{A,1}=(1, 8, 9)$, $\psi_6^{A,1}=(3,4,6)$, $(1,8,9)$, $\psi_6^{A,1}=(3,4,6)$, $\psi_7^{A,1}=(3,4,9)$, $\psi_8^{A,1}=(3,8,9)$ $\psi_9^{A,1}=(7,8,9)$. Among the 9 matches in $\Psi^{A,1}$, $\psi_1^{A,1}$, $\psi_3^{A,1}$, $\psi_6^{A,1}$, $\psi_9^{A,1}$ are minimal. $\mathbb{S}^2(A)$ has 6 matches in $\mathbb{E}$: $\psi_1^{A,2}=(1,2)$, $\psi_2^{A,2}=(1,4)$, $\psi_3^{A,2}=(1,8)$, $\psi_4^{A,2}=(3,4)$, $\psi_5^{A,2}=(3,8)$, $\psi_6^{A,2}=(7,8)$. $\psi_1^{A,2}$, $\psi_4^{A,2}$, $\psi_6^{A,2}$ are minimal.

[0079] In order to design efficient algorithms for solving the E2AP problem, attention is restricted to O(m) matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$ for each A and k, without losing important information. To achieve this goal, following concepts and solutions are provided.

[0080] Definition 2: (precedence relation on $\Psi^{A,k}$): A binary relation $\preceq$ on $\Psi^{A,k}$ is defined as follows. Let $\psi_i^{A,k}$ and $\psi_{i'}^{A,k}$ be two elements in $\Psi^{A,k}$. The term $\psi_i^{A,k} \preceq \psi_{i'}^{A,k}$ is true when either

$$\left(\psi_{i,|\mathbb{S}^k(A)|}^{A,k} < \psi_{i',|\mathbb{S}^k(A)|}^{A,k}; \text{ or } \psi_{i,|\mathbb{S}^k(A)|}^{A,k} = \psi_{i',|\mathbb{S}^k(A)|}^{A,k} \text{ and } \right. \quad \text{(i)}$$

$$\left(\psi_{i,|\mathbb{S}^k(A)|-1}^{A,k}, \psi_{i,|\mathbb{S}^k(A)|-2}^{A,k} \ldots \psi_{i,1}^{A,k}\right) \quad \text{(ii)}$$

is lexicographically greater than or equal to

$$\left(\psi_{i',|\mathbb{S}^k(A)|-1}^{A,k}, \psi_{i',|\mathbb{S}^k(A)|-2}^{A,k} \ldots, \psi_{i',1}^{A,k}\right).$$

[0081] When $\phi \preceq^{SM} \psi$ and $\phi \neq \psi$, the solutions states that ϕ precedes ψ, denoted by $\phi \prec \psi$. The solutions states that ϕ is lighter than ψ when $\phi \prec \psi$. One can verify that $\preceq$ defined above is a linear ordering on $\Psi^{A,k}$ (as well as on $\Psi^{min,A,k}$). In other words, the solutions states that:

[0082] 1) For any $\phi, \psi \in \Psi^{A,k}$, at least one in $\{\phi \preceq \psi, \psi \preceq \phi\}$ is true.

[0083] 2) If $\phi \preceq \psi$ and $\psi \preceq \phi$, then $\phi=\psi$.

[0084] 3) If $\phi \preceq \psi$ and $\psi \preceq \gamma$, then $\phi \preceq \gamma$.

[0085] Definition 3: (equivalence relation on $\Psi^{A,k}$ and $\Psi^{min,A,k}$): Let $A \in \mathcal{A}$, and $\mathbb{S}^k(A)$ be a possible pattern of A. Let $\psi_i^{A,k}$ and $\psi_{i'}^{A,k}$, be two elements in $\Psi^{A,k}$ ($\Psi^{min,A,k}$, respectively).

[0086] The solutions states that $\psi_i^{A,k}$ is equivalent to $\psi_{i'}^{A,k}$, denoted by $\Psi^{A,k} \equiv \psi_{i'}^{A,k}$, if the intervals of $\psi_i^{A,k}$ and $\psi_{i'}^{A,k}$ are the same.

[0087] Clearly, $\equiv$ is a binary relation defined on $\Psi^{A,k}$, as well as a binary relation defined on $\Psi^{min,A,k}$. One can verify that the binary relation $\equiv$ defined on $\Psi^{A,k}$ ($\Psi^{min,A,k}$, respectively) is an equivalence relation. This equivalence relation partitions the set $\Psi^{A,k}$ ($\Psi^{min,A,k}$ respectively) into equiva-

lence classes where two matches in $\Psi^{A,k}$ ($\Psi^{min,A,k}$, respectively) are in the same equivalence class if and only if they are equivalent.

[0088] Definition 4 (representative matches): Let $A \in \mathcal{A}$, and $\mathbb{S}^k(A)$ be a possible pattern of A. For each equivalence class of $\Psi^{A,k}$ ($\Psi^{min,A,k}$, respectively), the implementation chooses the lightest element in the class as its representative.

[0089] Example 3: Set A to A^2 in the example setting. There are 9 members in $\Psi^{A,1}$. In lexicographical order, they are (1, 2, 6), (1, 2, 9), (1, 4, 6), (1, 4, 9), (1, 8, 9), (3, 4, 6), (3, 4, 9), (3, 8, 9), (7, 8, 9). In lightest first order, there are (3, 4, 6)<(1, 4, 6)<(1, 2, 6)<(7, 8, 9)<(3, 8, 9)<(1, 8, 9)<(3, 4, 9)<(1, 4, 9)<(1, 2, 9).

[0090] The 5 equivalence classes of $\Psi^{A,1}$, with the representative of each class listed first in bold font, are {(3, 4, 6)}, {(1, 4, 6), (1, 2, 6)}, {(7, 8, 9)}, {(3, 8, 9), (3, 4, 9)}, and {(1, 8, 9), (1, 4, 9), (1, 2, 9)}. The 2 equivalence classes of $\Psi^{min,A,1}$ are {(3, 4, 6)} and {(7, 8, 9)}.

[0091] In Example 3, each equivalence class of $\Psi^{min,A,k}$ has only one match. In general, the number of elements in an equivalence class of $\Psi^{min,A,k}$ ($\Psi^{A,k}$) may be very large.

[0092] Lemma 1: Let $A \in \mathcal{A}$, and $\mathbb{S}^k(A)$ be a possible pattern of A. The number of equivalence classes of $\Psi^{A,k}$ is no more than m (m+1)/2. The number of equivalence classes of $\Psi^{min,A,k}$ is no more than m, where $m=|\mathbb{E}|$.

[0093] Proof of Lemma 1: By Definition 3, two matches of $\mathbb{S}^k(A)$ in $\mathbb{E}$ are equivalent if and only if their intervals are the same. Each interval may be represented in the form $[\alpha,\beta]$ where α and β are integers such that $1 \le \alpha \le \beta \le m$. The number of such intervals is m(m+1)/2. This proves the upper-bound on the number of equivalence classes of $\Psi^{A,k}$.

[0094] Let $\psi_i^{min,A,k}$ and $\psi_{i'}^{min}$, A, k be two non-equivalent elements of Ψ^{min}, A, k. Let $[\alpha,\beta]$ and $[\alpha',\beta']$ be the intervals of $\psi_i^{min,A,k}$ and $\psi_{i'}^{min}$, A, k, respectively. It is asserted that $\alpha \neq \alpha'$. Suppose to the contrary that $\alpha=\alpha'$. If $\beta=\beta'$, one may thus conclude that ψ_i^{min}, A, $k \equiv \psi_{i'}^{min}$, A, k; If $\beta<\beta'$, one may conclude that ψ_i^{min}, A, k is not minimal; If $\beta>\beta'$, one may conclude that ψ_i^{min}, A, k is not minimal. Therefore, it is proven that $\alpha \neq \alpha'$. This fact implies that the number of equivalence classes of $\Psi^{min,A,k}$ is upper-bounded by m.

Computing Representative Matches (Phase 1):

[0095] Described here is Phase 1 of the two-phase scheme as outlined in above. First presented is Algorithm 1 which may be used to compute a sequence $\mathbb{L}_{min,A,k}$ of the representatives for the equivalence classes of $\Psi^{min,A,k}$, for any given $A \in A$, and any $k \in \{0, 1, \ldots, |S(A)|$.

[0096] The data structures used in Algorithm 1 are as follows:

[0097] The term c[ ] is an integer-valued 2-D array of m+1 rows and $|\mathbb{S}^k(A)|+1$ columns. The entry c[i,j], when computed, is equal to $|LCS(\mathbb{E}(start:i), \mathbb{S}^k(A)(1:j))|$.

[0098] Here LCS(X,Y) denotes a longest common subsequence of X and Y, and |Z| denotes the length of sequence Z.

[0099] Integer l is used to index the next match $\psi_l \in \Psi^{min,A,k}$ found. At the end of the algorithm, l is the number of equivalence classes of $\Psi^{min,A,k}$.

[0100] $\mathbb{L}^{min,A,k}$ is a singly linked list, each node of which is an array of $|\mathbb{S}^k(A)|$ integers, corresponding to the $|\mathbb{S}^k(A)|$ indices of a match in $\Psi^{min,A,k}$. The operation $\mathbb{L}^{min,A,k}$ append appends a new node/match at the end.

[0101] The following is the flow of Algorithm 1:

[0102] Lines 1 and 4 perform initialization. In particular, begin by initializing c[start−1, j] to 0, which is |LCS (null, $\mathbb{S}^k(A)(1:j)$)| for all j and all start. Also, initialize c[i, 0] to 0, which is |LCS ($\mathbb{E}$(start:i), null)| for all i and all start.

[0103] In Line 1, Both start and i are set to 1 to get ready to compute the lightest minimal match of $\mathbb{S}^k(A)$ in $\mathbb{E}$ among those whose interval is contained in [start,m]. Line 20 also sets new values of start and i before control goes to Line 2 to start the computation of the next match in $\Psi^{min,A,k}$.

[0104] In consecutive executions of the while loop in Line 3 with the same start value, the aim is to compute the lightest minimal match of $\mathbb{S}^k(A)$ in E among those whose intervals are contained in [start,m], for the corresponding start value. Note that start is initialized to 1 in Line 1, and updated to a new (larger) value $\psi_l[1]+1$ in Line 20 after the l-th match is found. This update guarantees that the next match computed will be different from any of the previously computed matches.

[0105] For each fixed value of i, IoT learning framework **165** computes c[i,j] for j=1, 2, . . . , n in the for loop of Line 5. The condition in Line 6 is true if and only if $e_i = e_j^{A,k}$, indicating that IoT learning framework **165** just found a matched pair of events. Due to this match, IoT learning framework **165** has $|LCS(\mathbb{E}(start:i), \mathbb{S}^k(A)(1:j))| = 1+|LCS(\mathbb{E}(start:i-1), \mathbb{S}^k(A)(1:j-1))|$. Therefore in Line 7, IoT learning framework **165** sets c[i,j]←c[i−1, j−1]+1=|LCS ($\mathbb{E}$(start:i−1), $\mathbb{S}^k(A)(1:j-1)$)|+1=|LCS($\mathbb{E}$(start:i), $\mathbb{S}^k(A)(1:j)$)|. Otherwise, c[i,j] is either set to c[i−1, j] in Line 9 or c[i,j−1] in Line 10 to guarantee c[i,j]=|LCS ($\mathbb{E}$(start:i), $\mathbb{S}^k(A)(1:j)$)|.

[0106] In Line 11, the algorithm checks whether a match of $\mathbb{S}^k(A)$ in $\mathbb{E}$(start:i) is found. If this condition is not true, control jumps to Line 21 where i is incremented and the statement of the while loop is executed again if the new value of i does not exceed m, and the algorithm outputs $\mathbb{L}^{min,A,k}$ if the new value of i exceeds m. If the condition in Line 11 is true, the algorithm backtracks to trace out the newly found match of $\mathbb{S}^k(A)$ in $\mathbb{E}$(start:i) in Lines 12 to 19.

[0107] In Line 20, the algorithm updates the values of start and i to i to $\psi_l[1]+1$ and $\psi_l[1]$, respectively. In Line 2, the algorithm initializes the entries of c[ ] in row start−1 to 0 to prepare for the computation of the lightest minimal match of $\mathbb{S}^k(A)$ in $\mathbb{E}$(start: m). The previous values in the overwritten rows are no longer needed.

[0108] FIG. **5** sets forth Table 2 at element **501**, depicting an example graphical illustration of the AkMatch functionality, in accordance with aspects of the disclosure.

[0109] Example 4: The example setting was used to illustrate the execution of AkMatch ($\mathbb{E}$, A^2, 1), with the aid of Table 2 (**501**).

[0110] Pattern $\mathbb{S}^1(A^2)$=(a, b, c) is on top, and sequence $\mathbb{E}$=(a, b, a, b, d, c, a, b, c) is on the left. To make things clear, $start_1$, start 2, and $start_3$ are used to denote the different start values. For an entry that is written more than once, comma (s) are utilized to separate these values, with newer values towards the left.

[0111] First, IoT learning framework **165** sets start to 1 (denoted by start 1) and initialize c[0, j] to 0 for 0≤j≤3. For i=1, IoT learning framework **165** has c[1,0]=0,

$$c[1,1]=1\left(\text{since } e_1=e_1^{A^2,1}=a\right),$$

c[1,2]=1, c[1,3]=1. Similarly, IoT learning framework **165** has c[2,0]=0, c[2,1]=1; c[2,2]=2, c[2,3]=2; c[3,0]=0, c[3,1]=1; c[3,2]=2, c[3,3]=2; c[4,0]=0, c[4,1]=1; c[4,2]=2, c[4,3]=2; c[5,0]=0,c[5,1]=1; c[5,2]=2, c[5,3]=2; c[6,0]=0, c[6,1]=1; c[6,2]=2, c[6,3]=3. Now IoT learning framework **165** has c[i, 3]=3, with i=6. This indicates there is a match of $\mathbb{S}^1(A^2)$ in E that lies within $\mathbb{E}(1{:}6)$.

[0112] The algorithm backtracks from cell [6, 3] to trace out the lightest match of $\mathbb{S}^1(A^2)$ in E that lies within E (**1**:**6**). As shown here, the algorithm shades the cells on the backtracking path and adds a frame-box around the value of c[row, col] if the condition in Line 14 is true, and adds a gray background if the condition in Line 16 is true. In cell [6, 3], the algorithm sets ψ_1 [3]→6, and move to cell [5, 2]. In cell [5, 2], the algorithm moves to cell [4, 2]. In cell [4, 2], the algorithm sets ψ_1 [2]→4, and move to cell [3, 1]. In cell [3, 1], the algorithm sets ψ_1 [1]→3, and move to cell [2, 0]. This backtrack stops at cell [2, 0]. By now, the algorithm has found

$$\psi_1^{min,A^2,1}=(3,4,6).$$

Note that there may be multiple matches of $\mathbb{S}^1(A^2)$ in $\mathbb{E}$ that lie entirely in $\mathbb{E}(1{:}6)$. The match (3, 4, 6) computed by the algorithm is minimal and the lightest among these matches. In the example, (1, 4, 6) and (1, 2, 6) also lie within $\mathbb{E}(1{:}6)$. As illustrated in Example 3, (3, 4, 6) is minimal and is lighter than both (1, 4, 6) and (1, 2, 6).

[0113] Next, the algorithm sets start

$$\leftarrow \psi_1^{min,A^2,1}[1]+1=4$$

and start the computation of the next match. Some of the cells will be overwritten by the algorithm with values computed with a new start value. The algorithm finds c[9, 3]=3, and backtracks to find the next match

$$\psi_2^{min,A^2,1}=(7,8,9),$$

which is the only match of $\mathbb{S}^1(A^2)$ in $\mathbb{E}$ that lies entirely in $\mathbb{E}(4{:}9)$.

[0114] Then, the algorithm sets start

$$\leftarrow \psi_2^{min,A^2,1}[1]+1=8,$$

starts the computation of the next match. This time, the algorithm could not find a match. In summary, the algorithm finds two matches of $\mathbb{S}^1(A^2)$ in

$$\mathbb{E}\text{: }\psi_1^{min,A^2,1}=(3,4,6) \text{ and } \psi_2^{min,A^2,1}=(7,8,9).$$

[0115] Theorem 1: For given $\mathbb{E}$, A, and k, Algorithm 1 computes list $\mathbb{L}^{min,A,k}$ of all representatives of equivalence classes of $\Psi^{min,A,k}$, in increasing order defined by ordering $\prec$. The worst-case running time of the algorithm is $O(m^2|\mathbb{S}^k(A)|)$.

[0116] Each match in $\Psi^{A,k}$ can be used as a possible match of the pattern $\mathbb{S}^k(A)$ of user activity A. In order to avoid double-counting of a device event, the following concept is introduced, according to Definition 5:

[0117] Definition 5 (compatible matches): Let $\psi_i^{A,k}$ be a match in $\Psi^{A,k}$ for A∈ $\mathbb{E}$ $\mathcal{A}$ and k$\mathbb{E}$ {1, $|\mathcal{S}(A)|$}. Let $\psi_{i'}^{A',k'}$ be a match in $\Psi^{A',k'}$ for A'∈ $\mathcal{A}$ and k'∈{1, $|\mathcal{S}(A')|$}. It is stated that $\psi_i^{A,k}$ and $\psi_{i'}^{A',k'}$ are compatible if the intervals of $\psi_i^{A,k}$ and $\psi_{i'}^{A',k'}$ do not overlap according to Equation 4, set forth below, as follows:

$$\psi_{i,1}^{A,k}>\psi_{i',|\mathbb{S}^k(A')|}^{A',k'} \text{ or } \psi_{i,|\mathbb{S}^k(A)|}^{A,k}<\psi_{i',1}^{A',k'}.$$

[0118] Let $\Psi=\cup_{A\in\mathcal{A},1\le k\le|\mathcal{S}(A)|}\Psi^{A,k}$ be the set of all matches of a pattern $\mathbb{S}^k(A)$ in $\mathbb{E}$, for some user activity A∈ $\mathcal{A}$.

[0119] Assume that W>0 is a given real number and that for each i=1, 2, . . . , $|\mathcal{A}|$ and each k=1, 2, . . . , $|\mathcal{S}(A^i)|$, there is a nonnegative real number w(i,k)≥0 such that $\sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{|\mathcal{S}(A)|}$ w(i,k)=W. For each $\psi_j^{A^i,k}\in\Psi$, IoT learning framework **165** associates a weight $\psi_j^{A^i,k}$.weight=w(i,k). Let Φ⊆Ψ be a subset of Ψ. The weight

$$w(\Phi)=\sum_{\psi_j^{A^i,k}\in\Phi}w(i,k).$$

Φ is said to be compatible if the elements in Φ are mutually compatible.

[0120] Proof of Theorem 1: Algorithm 1 is a modification of the algorithm for computing an LCS of two sequences $\mathbb{E}$(start: m) and $\mathbb{S}^k(A)$. The difference lies in (i) only of interest is the computation of an LCS that is identical to $\mathbb{S}^k(A)$, not any proper subsequence of $\mathbb{S}^k(A)$; and (ii) multiple matches are computed, rather than one. Therefore the details that are well-known in the correctness of the algorithm for LCS are omitted.

[0121] The term n is used to denote $|\mathbb{S}^k(A)|$. Computing one LCS requires O(mη) time in the worst-case. The algorithm computes O(m) LCSs. Hence the worst-case running time is $O(m^2\eta)$.

[0122] If LCS ($\mathbb{E}$, $\mathbb{S}^k(A)$)≠$\mathbb{S}^k(A)$, Algorithm 1 outputs a null list, as $\Psi^{min,A,k}=\Psi^{A,k}=\emptyset$ in this case. In the rest of the proof, it may be assumed that LCS ($\mathbb{E}$, $\mathbb{S}^k(A)$)=$\mathbb{S}^k(A)$. Hence the condition in Line 11 is true at least once during the execution.

[0123] Each time when the condition in Line 11 is true, it is known that $\mathbb{E}$ (start:i−1) contains no subsequence that is identical to $\mathbb{S}^k(A)$, and that $\mathbb{E}$(start:i) contains a subsequence that is identical to $\mathbb{S}^k(A)$. Therefore Lines 12-19 correctly compute a match ψ_l=($\psi_l[1]$, $\psi_l[2]$, . . . , $\psi_l[\eta]$) of $\mathbb{S}^k(A)$ in $\mathbb{E}$(start:i).

[0124] Let x15 and x18 denote the number of times Line 15 and Line 18 are executed (with start and l fixed), respectively. Since control enters the while loop in Line 13

with col initialized to η, and exits the while loop when col is reduced to 0, IoT learning framework **165** has x15+x18=η. Because a match of $\mathbb{S}^k(A)$ in $\mathbb{E}$ is found that lies entirely in $\mathbb{E}$(start: i), IoT learning framework **165** has x15=η. Hence x18=0. In other words, between the executions of Line 12 and Line 19, Line 15 is executed exactly η times, and Line 18 is executed zero times.

[0125] When IoT learning framework **165** traces out a newly computed match of $\mathbb{S}^k(A)$ in $\mathbb{E}$, IoT learning framework **165** first initializes row to i and col to η. If $e_{row}=e_{col}^{A,k}$, IoT learning framework **165** decrements both row and col by 1. Otherwise, IoT learning framework **165** decrements row by 1 and keep col unchanged. Therefore, the match IoT learning framework **165** traces out is the lightest among all matches of $\mathbb{S}^k(A)$ in E that lie entirely in $\mathbb{E}$(start,i).

[0126] When the condition in Line 11 becomes true for the first time, IoT learning framework **165** may determine that there is no match of $\mathbb{S}^k(A)$ in E that lies within $\mathbb{E}$(start:i)=$\mathbb{E}$(1: i). Therefore $\psi_1=(\psi_1[1], \psi_1[2], \ldots, \psi_1[\eta])$ is the lightest match of $\mathbb{S}^k(A)$ in $\mathbb{E}$, which implies that it is minimal. Let Let $\phi=(\phi[1], \phi[2], \ldots, \phi[\eta])$ be any minimal match of $\mathbb{S}^k(A)$ in $\mathbb{E}$ that is not equivalent to ψ_1, IoT learning framework **165** may have $\phi[1]\geq\psi_1[1]+1$. Therefore ψ_2 computed by IoT learning framework **165** using the algorithm is the lightest minimal match of $\mathbb{S}^k(A)$ in $\mathbb{E}$ that is not equivalent to ψ_1. Similarly, ψ_3 computed by the algorithm is the lightest minimal match of $\mathbb{S}^k(A)$ in E that is not equivalent to ψ_1 or ψ_2. In general, ψ_1 computed by the algorithm is the lightest minimal match of $\mathbb{S}^k(A)$ in $\mathbb{E}$ that is not equivalent to ψ_j for j=1, 2, . . . , l−1. This completes the proof of the theorem.

[0127] The following optimization problem is studied below according to Problem 3. Problem 3 (MaxWCM (Φ,w)): Let W>0 be given, together with weights w (i,k)≥0 for i=1, 2, . . . , $|\mathcal{A}|$ and k=1, 2, . . . $|\mathcal{S}(A^i)|$ such that $\sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{|\mathcal{S}(A^2)|}$w(i,k)=W. Let Φ be a subset of Ψ. The weighted maximum compatible match selection problem (MaxWCM) on (Φ,w) seeks for a maximum weight compatible subset $\Phi_{w,opt}\subseteq\Phi$.

[0128] Algorithm 1 may be applied to compute the list $\mathbb{L}^{min,A^2,k}$ of matches in $\Psi^{min,A^i,k}$ for i=1, . . . , $|\mathcal{A}|$, k=1, . . . , $|\mathcal{S}(A^i)|$.

[0129] Let $\overline{\Psi}=\bigcup_{A^i\in\mathcal{A},1\leq k\leq|\mathcal{S}(A)}\{\psi_j^{A^i,k}|\psi_j^{A^i,k}\in\mathbb{L}^{min,A^i,k}\}$. The following theorem shows that an optimal solution of the Max WCM problem on ($\overline{\Psi}$,w) is also an optimal solution of the MaxWCM problem on (Ψ,w). Note that $\overline{\Psi}$ is a subset of Ψ whose cardinality is much smaller than that of Ψ.

[0130] Theorem 2: The MaxWCM(Ψ,w) problem has an optimal solution $\Psi_{w,opt}\subseteq\overline{\Psi}$. Furthermore, such an optimal solution can be obtained by solving the MaxWCM($\overline{\Psi}$,w) problem.

[0131] Note that Ψ contains every match of $\mathbb{S}^{A^2,k}$, for every $A^i\in\mathcal{A}$ and every $k\in\{1\leq k\leq|\mathcal{S}(A^i)|\}$. Ψ^{min} is a small subset of Ψ, consisting of the minimal matches only. $\overline{\Psi}$ is a small subset of 4 min, consisting of the representatives of its equivalence classes. Hence $|\overline{\Psi}|$ is much smaller than |Ψ| in general. Theorem 2 shows that IoT learning framework **165** does not lose any important information by diverting attention away from the very large set Ψ to the much smaller set $\overline{\Psi}$.

[0132] Proof of Theorem 2: Let $\Psi^{min}=\bigcup_{A^i\in\mathcal{A},1\leq k\leq|\mathcal{S}(A^i)|}\Psi^{min,A^i,k}$. First it is proven that there is an optimal solution $\Psi_{w,opt}$ of the MaxWM(Ψ,w) problem such that $\Psi_{w,opt}\subseteq\Psi^{min}$. Let $\Psi_{w,opt}$ be an arbitrary optimal solution for the MaxWCM(Ψ,w) problem. If $\Psi_{w,opt}\subseteq\Psi^{min}$, there is nothing to be proved. Let $\psi_j^{A^i,k}$ be a match in $\Psi_{w,opt}$ such that $\psi_j^{A^i,k}\in\Psi^{A^i,k}\dagger\Psi^{min,A^i,k}$. Let $\psi_{j'}^{A^i,k}$ be a minimal match of $\mathbb{S}(A^i)$ in $\mathbb{E}$ such that $[\psi_j^{A^i,k}[1], \psi_{j'}^{A^i,k}[|\mathbb{S}^k(A^i)|]$ is a proper subinterval of $[\psi_j^{A^i,k}[1], \psi_j^{A^i,k}[|\mathbb{S}^k(A^i)|]$. Since the two matches have the same weight w(i,k), IoT learning framework **165** may obtain another optimal solution by replacing $\psi_j^{A^i,k}$ with $\psi_{j'A}^{i,k}$. Therefore, IoT learning framework **165** may obtain another optimal solution to the MaxWCM(Ψ,w) problem with one fewer non-minimal match than $\Psi_{w,opt}$. Repeating this process for each non-minimal match in $\Psi_{w,opt}$, IoT learning framework **165** may obtain an optimal solution to the MaxWCM(Ψ,w) problem consisting of matches in Ψ^{min} only.

[0133] For WLOG, assume that $\Psi_{w,opt}\subseteq\Psi^{min}$ in the rest of this proof. Next, it is shown that the optimal solution $\Psi_{w,opt}$ can be transformed to another optimal solution $\Psi_{w,opt}\subseteq\overline{\Psi}$.

[0134] Suppose there is an element $\psi_j^{min,A^i,k}\in\Psi_{w,opt}$ that is not in $\overline{\Psi}$. Let $\psi_{j_{lightest}}^{min,A^i,k}$ be the lightest element in the equivalence class which contains $\psi_j^{min,A^i,k}$. In such an event, $\psi_{j_{min}}^{min,A^i,k}$ can be replaced with $\psi\psi_{j_{lightest}}^{min,A^i,k}$. This transformation does not destroy compatibility (since $\psi_{j_{lightest}}^{min,A^i,k}$ and $\psi_j^{min,A^i,k}$ have the same interval), nor does it change the weight of the set (since $\psi_{j_{lightest}}^{min,A^i,k}$ and $\psi_j^{min,A^i,k}$ have the same weight w (i,k)). Note that $\psi_{j_{lightest}}^{min,A^i,k}\in\overline{\Psi}$. Therefore, through a finite number of such transformations, IoT learning framework **165** can transform the optimal solution $\Psi_{w,opt}$ to another optimal solution $\Psi_{w,opt}\subseteq\overline{\Psi}$.

[0135] FIG. **6** sets forth Algorithm 2 at element **601**, depicting an example implementation of the OMatch($\mathbb{M}$,w) functionality for computing a compatible sub-sequence of user activity patterns, in accordance with aspects of the disclosure.

Inferring Optimal Sequence of User Activity Patterns with a Given Weight (Phase 2A):

[0136] In this section, an efficient algorithm is presented for solving the MaxWCM problem on ($\overline{\Psi}$,w) for any given weight w. For ease of presentation, a uniform representation of the matches in $\overline{\Psi}$ is utilized, explained in the following.

[0137] Each match $\psi_j^{min,A^2,k}\in\overline{\Psi}$ is characterized by a 5-tuple (i,k, α,β,w), where

$$\alpha=\psi_{j,1}^{min,A^i,k}=\psi_j^{min,A^i,k}, \beta=\psi_{j|\mathbb{S}^k(A^i)|}^{min,A^i,k}=\psi_j^{min,A^i,k}[|\mathbb{S}^k(A^i)|],$$

and $\psi_j^{min,A^i,k}$ is the weight w(i,k) associated with $\mathbb{S}^k(A^i)$.

[0138] Let $m_{i,k}=|\mathbb{L}^{min,A^i,k}|$, i=1, . . . , $|\mathcal{A}|$, k=1, . . . $|\mathcal{S}(A^i)|$. Let $M=\sum_{i=1}^{|\mathcal{A}|}\sum_{k=1}^{|\mathcal{S}(A^2)|}m_{i,k}$. Let $\mathbb{M}$ be a 1-D array of M 5-tuples (i,k, α,β,w), each of which corresponds to a match computed. The array $\mathbb{M}$ can be sorted according to nondecreasing value of β in O(M log M) time. Without loss of generality, it may be assumed that array $\mathbb{M}$ is already sorted. It may further be assumed that IoT learning framework **165** has computed a 1-D array p of M+1 integers where p[0]=0 and p[j] is the largest integer t∈{0, 1, . . . , j−1} such that $\mathbb{M}$[t]. β< $\mathbb{M}$[j]. α. Here the technical convention that

$\mathbb{M}[0]\cdot\beta=-\infty$ is utilized. Algorithm 2 provides a solution for the MaxWCM(Ψ,w) problem.

[0139] Theorem 3: Algorithm 2 computes a compatible subsequence $\mathbb{M}_w$ of user activity patterns in $\mathbb{M}$ with maximum total weight. Its worst-case running time is O(M).

[0140] IoT learning framework **165** may implement a solution to E2AP and E2A utilizing the following flow. For each $A^i\in\mathcal{A}$, i=1, 2, . . . , $|\mathcal{A}|$, IT learning framework **165** may determine the possible patterns $\mathcal{S}(A^i)=\{\mathbb{S}^1(A^i), \mathbb{S}^2(A^i), \ldots, \mathbb{S}^{|\mathcal{S}(A^i)|}(A^i)\}$. Given the sequence $\mathbb{E}$ of device events, by repeated applications of Algorithm 1 for each $A^i\in\mathcal{A}$, i=1, 2, . . . , $|\mathcal{A}|$ and each $k\in\{0, 1,2, \ldots, |\mathcal{S}(A^i)|\}$, IoT learning framework **165** can compute $\mathbb{L}^{min,A^i,k}$ of representative matches in $\Psi^{min,A^i,k}$ for each i and k. The union of these lists is Ψ. Given the weights w(i,k), IoT learning framework **165** can apply Algorithm 2 to compute $\mathbb{M}_w$. From $\mathbb{M}_w$, IT learning framework **165** may obtain a sequence of $|\mathbb{M}_w|$ user activity patterns $\mathbb{S}^{w\,w}=(\mathbb{S}^{w\,w}[1], \mathbb{S}^{w\,w}[2], \ldots, \mathbb{S}^{w\,w}[|\mathbb{M}_w|])$, where $\mathbb{S}^{w\,w}[j]=\mathbb{S}^{A^{\mathbb{M}w[j]\cdot i},\mathbb{M}_w[j]\cdot k}$, j=1, 2, . . . , $|\mathbb{M}_w|$. Ignoring the patterns, IoT learning framework **165** may obtain a sequence of $|\mathbb{M}_w|$ user activities $\mathbb{A}^{w\,w}(\mathbb{A}^{w\,w}[1], \mathbb{A}^{w\,w}[2], \ldots, \mathbb{A}^{w\,w}[|\mathbb{M}_w|])$, where $\mathbb{A}^{w\,w}$ $A^{\mathbb{M}w[j]\cdot i}$, j=1, 2, . . . , $|\mathbb{M}_w|$. IoT learning framework **165** may use $\mathbb{S}^{w\,w}$ and $\mathbb{A}^{w\,w}$ for solving for E2AP and E2A, respectively.

[0141] Example 5: Let $A=(A^1, A^2, A^2)$ be a sequence of user activities, where A^1 and A^2 are as in the example setting. Applying the AkMatch algorithm, IT learning framework **165** obtains $\mathbb{L}^{min,A^1,1}=((1,2), (3,4), (7,8))$, $\mathbb{L}^{min,A^2,1}=((3,4, 6), (7,8,9))$, and $\mathbb{L}^{min,A^2,2}=((1,2), (3,4), (7,8))$. Putting these 8 matches into array $\mathbb{M}$ and sorting according to the β field, IoT learning framework **165** has $\mathbb{M}[1]=(1,1,1,2, w(1,1))$, $\mathbb{M}[2]=(2,2,1,2, w(2,2))$, $\mathbb{M}[3]=(1,1,3,4, w(1,1))$, $\mathbb{M}[4]=(2, 2,3,4, w(2,2))$, $\mathbb{M}[5]=(2,1,3,6, w(2,1))$, $\mathbb{M}[6]=(1,1,7,8, w(1, 1))$, $\mathbb{M}[7]=(2,2,7,8, w(2,2))$, $\mathbb{M}[8]=(2,1,7,9, w(2,1))$. IoT learning framework **165** also has p[0]=p[1]=p[2]=0, p[3]=p [4]=p[5]=2, p[6]=p[7]=p[8]=5.

[0142] Suppose w (1,1)=1.4, w(2,1)=1.6, w(2,2)=0. The OMatch algorithm will compute $\mathbb{M}_w=(\mathbb{M}[1], \mathbb{M}[5], \mathbb{M}[8])$.

[0143] Note that $\mathbb{M}[1]$ corresponds to $\mathbb{S}^1(A^1)$, $\mathbb{M}[5]$ corresponds to $\mathbb{S}^1(A^2)$, and $\mathbb{M}[8]$ corresponds to $\mathbb{S}^1(A^2)$. This leads to the sequence of user activity patterns Sw=(S1 (A1), S1(A2), S1(A2)) as a solution to E2AP and the sequence of user activities Aw=(A1, A2, A2) as a solution to E2A. Note that the edit distance between $\mathbb{E}$ and $\mathbb{S}^1(A^1)\|\mathbb{S}^1(A^2)\|\mathbb{S}^1(A^2)$ is 1.

[0144] Suppose w(1,1)=1.6, w(2,1)=1.4, w(2,2)=0. The OMatch algorithm will compute $\mathbb{M}_w=(\mathbb{M}[1], \mathbb{M}[3], \mathbb{M}[6])$ This leads to the sequence of user activity patterns $\mathbb{S}^w=(\mathbb{S}^1(A^1), \mathbb{S}^1(A^2), \mathbb{S}^1(A^2))$ as a solution to E2AP and the sequence of user activities $\mathbb{A}^w=(A^1, A^2, A^2)$ as a solution to E2A. Note that the edit distance between $\mathbb{E}$ and $\mathbb{S}^1(A^1)\|\mathbb{S}^1(A^1)\|\mathbb{S}^1(A^1)$ is 3.

[0145] Example 5 shows that the value of the weights has a big impact on the quality of $\mathbb{S}^w$ ($\mathbb{A}^w$, respectively) as a solution to E2AP (E2A, respectively). In Phase 2B, IoT learning framework **165** utilizes unsupervised learning to compute a good weight, by minimizing a properly defined loss function.

[0146] Proof of Theorem 3: Algorithm 2 is the dynamic programming algorithm for weighted interval scheduling.

Learning Optimal Weights (Phase 2B):

[0147] As described above, w uniquely decides $\mathbb{M}_w$, which in turn uniquely decides $\mathbb{S}^w$ and $\mathbb{A}^w$, as the solution to E2AP and E2A, respectively. In general, $\mathbb{S}^w$ and $\mathbb{S}^{w'}$ ($\mathbb{A}^w$ and $\mathbb{A}^{w'}$, respectively) are not equally good solutions to E2AP (E2A, respectively) when w≠w', as illustrated in Example 5.

[0148] In this section, an unsupervised learning approach is presented via which to learn a proper weight assignment. Both supervised and unsupervised learning can be used. Since the E2AP problem asks for activity patterns rather than activities, a loss function can be defined that makes unsupervised learning more adaptive. Hence the focus is on unsupervised learning. The application of supervised learning as well as distributed unsupervised learning to this problem may similarly be leveraged as alternative or additional expansions to the methodologies described herein.

[0149] Provided below, the loss function is defined as the edit distance between $\mathbb{E}$ and the concatenation of the activity patterns computed. The loss function is a non-differentiable function of the continuous variables corresponding to the weights. Still further, the exact algorithm to minimize the loss function with all but one variable fixed is defined and implemented as described. This algorithm is used as a sub-routine to design a coordinated descent algorithm to minimize the loss function.

Loss Function and Optimization Problem Formulation:

[0150] Let $\mathbb{E}^w$ be the sequence of device events obtained by concatenating the activity patterns in $\mathbb{S}^w$, according to Equation 5, set forth below, as follows:

$$\mathbb{E}^w=\mathbb{S}^w[1]\|\mathbb{S}^w[2]\|\ \ldots\ \|\mathbb{S}^w[|\mathbb{M}_w|].$$

[0151] Without any knowledge of the ground truth, the loss function for parameter w is defined as the edit distance between $\mathbb{E}^w$ and $\mathbb{E}$, denoted by $d(\mathbb{E}^w, \mathbb{E})$. Here it is assumed that the operations deletion, insertion, and substitution all have costs equal to 1.

[0152] Let $n=|\mathcal{A}|$; $n_j=|\mathcal{S}(A^j)|$ for j=1, 2, . . . , n; and $N=\Sigma_{j=1}^{n} n_j$. Then w consists of N variables. A natural approach to obtaining the optimal values of w is to solve the following optimization problem according to Equation 6, set forth below as follows:

$$\underset{w}{\text{minimize}} f(w)=d(\mathbb{E}^w, \mathbb{E}),$$

where Equation 6 is subject to Equation 6A, set forth as follows:

$$\sum_{i=1}^{n}\sum_{k=1}^{|\mathcal{S}(A^i)|} w(i,k)=N,$$

and further subject to Equation 6B, set forth as follows:

$$w(i,k) \geq 0, \forall\, i = 1, \ldots, n, \forall\, k = 1, \ldots, |\mathcal{S}(A^i)|.$$

[0153] Equation 6 and its sub-parts are difficult to solve, as the objective function is non-differentiable and non-convex. In an effort to design a coordinated descent algorithm for solving Equation 6, an optimization problem is evaluated to minimize the objective function over one of the N variables. This problem may be formally defined in the following, where w(i,k) is the variable, for a chosen pair (i,k), according to Equation 7, set forth below as follows:

$$\underset{w(i,k)}{\text{minimize}} f(w) = d(\mathbb{F}^w, \mathbb{F}),$$

subject to Equation 7A, set forth as follows:

$$w(i,k) \geq 0.$$

[0154] Equation 7 may be referred to as the 1-D problem, and Equation 6 may be referred to as the N-D problem. An exact algorithm is provided for solving Equation 7 and a coordinate descent algorithm is provided for solving Equation 6.

[0155] FIGS. **7**A, **7**B, and **7**C set forth Algorithm 3 at elements **701**A, **701**B, and **701**C depicting an example implementation of the SLN-1D functionality for computing an optimal solution to the 1-D problem, in accordance with aspects of the disclosure.

Exact Algorithm for 1-D Optimization:

[0156] An algorithm named SLN-1D enables IoT learning framework **165** to compute an optimal solution of the 1-D problem, and list it as Algorithm 3. Algorithm 3 performs many rounds of Algorithm 2, by tracking the trend of the computed matches with the value of w(i,k) taking on larger and larger values. Here $\mathbb{M}$ and p are the same as in Algorithm 2. The key observation behind the design of Algorithm 3 is that the objective function f(w) in Equation 7 is a step function of w(i,k) with a finite number of different function values, each of which is defined on an interval for w(i,k). IoT learning framework **165** can evaluate f(w) in an interior point and on the boundaries of each of these intervals. IoT learning framework **165** may find the boundaries of these intervals by performing a left-to-right scan. IoT learning framework **165** may start the scan at w(i,k)=0 (Line 1). For each value x of w(i,k), IoT learning framework **165** computes real numbers μ and ν where μ<x and

$$v = \frac{x+\mu}{2}$$

such that (i) $\mathbb{M}^w$ does not change when w(i,k) is varying in the interval (x, μ) and (ii) such that $\mathbb{M}^w$ will change when w(i,k) is increased from x to μ or from x to μ plus an infinitesimal amount.

[0157] IoT learning framework **165** evaluates f(w) by setting w(i,k) to ν and μ, respectively. Then repeat the process with x set to μ, as long as such a μ exists. When such a μ does not exist, IoT learning framework **165** sets μ→x+2 and ν→x+1, and evaluate f(w) by setting w(i,k) to ν and μ, respectively. Lines 2-25 carry out the computation of w(i,k) for the current value of x. In Line 26, IoT learning framework **165** sets u→x+σ and ν→x+σ/2. Lines 27-35 perform the function evaluations at ν and μ, and related bookkeeping operations.

[0158] The computation of σ=μ−x from x is carried out by operations similar to the operations in the dynamic programming algorithm for weighted interval scheduling, e.g., Lines 1-6 of Algorithm 2. There are additional operations needed to compute σ.

[0159] Before proceeding with the explanation of the algorithm, the meaning of two variables are described: array δ[ ] and array a[ ]. For j=1, 2, . . . , M, δ[j]=1 if $\mathbb{M}[j]\cdot w$ is the variable weight w(i,k), δ[j]=0 otherwise. For j=1, 2, . . . , M, IoT learning framework **165** may use a [j] to denote the number of selected matches/intervals in the optimal solution of $\mathbb{M}$ (1:j) whose weight is w(i,k), when w(i,k) is set to x plus an infinitesimal amount.

[0160] In Line 2, IT learning framework **165** sets OPT[0] →0, a [0]→0, σ→∞. The for loop in Lines 3-24 computes the values for δ[j], OPT[j], α[j], and reduces σ accordingly. The body of the for loop starts with the computation of δ[j] in Lines 4-7. This is followed by three mutual exclusive parts: Lines 9-12 (when $\mathbb{M}[j]\cdot w+OPT[p[j]]>OPT[j-1]$), Lines 14-17 (when $\mathbb{S}^w$ $[j]\cdot w+OPT[p[j]]<OPT[j-1]$), and Lines 19-24 (when $\mathbb{M}[j]\cdot w+OPT[p[j]]=OPT[j-1]$).

[0161] When $\mathbb{M}[j]\cdot w+OPT[p[j]]>OPT[j-1]$, the optimal solution of $\mathbb{M}(1:j)$ consists of $\mathbb{M}[j]$ and the optimal solution of $\mathcal{A}(1:p[j])$. This will not change even when w(i,k) is increased by an infinitesimal amount, due to the strict inequality. Hence a [j] is computed according to Line 10. If $\alpha[p[j]]+\delta$ $[j]<\alpha[j-1]$, the inequality $\mathbb{M}[j]\cdot w+OPT[p[j]] >OPT[j-1]$ will no longer be true when w(i,k) is increased by an amount equal to

$$\frac{\mathbb{M}[j]\cdot w + OPT[p[j]] - \bar{O}PT[j-1]}{a[j-1] - a[p[j]] - \delta[j]}.$$

Therefore, IoT learning framework **165** sets σ to the smaller of its current value and

$$\frac{\mathbb{M}[j]\cdot w + OPT[p[j]] - OPT[j-1]}{a[j-1] - 1[p[j]] - \delta[j]}$$

in Line 12.

[0162] When $\mathbb{M}[j]\cdot w+OPT[p[j]]<OPT[j-1]$, the optimal solution of $\mathbb{M}(1:j)$ is the optimal solution of $\mathbb{M}(1:j-1)$. This will not change even when w(i,k) is increased by an infinitesimal amount, due to the strict inequality. Hence a [j] is computed according to Line 15. If $\alpha[p[j]]+\delta[j]>\alpha[j-1]$, IoT learning framework **165** will have $\mathbb{M}[j]\cdot w+OPT[p[j]]>OPT$ [j−1] when w(i,k) is increased by an amount equal to an infinitesimal plus

$$\frac{\mathbb{M}[j]\cdot w + OPT[p[j]] - OPT[j-1]}{a[j-1] - 1[p[j]] - \delta[j]}.$$

Therefore, IoT learning framework **165** sets σ to the smaller of its current value and

$$\frac{\mathbb{M}[j]\cdot w + OPT[p[j]] - OPT[j-1]}{a[j-1] - 1[p[j]] - \delta[j]}$$

in Line 17.

[0163] When $\mathbb{M}[j]\cdot w+OPT[p[j]]=OPT[j-1]$, the optimal solution of $\mathbb{M}(1:j)$ is the optimal solution of $\mathbb{M}(1:j-1)$, with an objective function value equal to OPT[j−$\mathbb{M}$[j]. w+OPT[p[j]]. If a $[p[j]]+\delta[j]\leq\alpha[j-1]$, IoT learning framework **165** will have $\mathbb{M}[j]\cdot w+OPT[p[j]]\leq OPT[j-1]$ when w(i,k) is increased by an infinitesimal amount. Note that $\mathbb{M}[j]\cdot w+OPT[p[j]]\leq OPT[j-1]$ implies that the optimal solution for $\mathbb{M}(1:j)$ is the optimal solution for $\mathbb{M}(1:j-1)$. Therefore, IoT learning framework **165** may select the assignment of OPT[j] in Line 20 and set the value of a [j] according to Line 21. If a $[p[j]]+\delta[j]>\alpha[j-1]$, IoT learning framework **165** will have $\mathbb{M}[j]\cdot w+OPT[p[j]]>OPT[j-1]$ when w(i,k) is increased by an infinitesimal amount. Note that $\mathbb{M}[j]\cdot+OPT[p[j]]>OPT[j-1]$ implies that the optimal solution for $\mathbb{M}(1:j)$ consists of $\mathbb{M}[j]$ and the optimal solution for $\mathbb{M}(1: p[j])$. Therefore, IoT learning framework **165** may select the assignment of OPT[j] in Line 23 and set the value of a [j] according to Line 24.

[0164] If IoT learning framework **165** finds σ=∞ in Line 25, IoT learning framework **165** has identified that x is the left end of the rightmost interval for w(i,k). However, IoT learning framework **165** may not have determined whether the function is right-continuous at x. Therefore, IoT learning framework **165** sets σ to 2 (which can be replaced by any other positive number). In Line 26, IoT learning framework **165** sets μ←x+σ and ν←x+0.5σ. Lines 27-35 are straightforward which do not need explanation. If the current x is not the left end of the rightmost interval, IoT learning framework **165** sets x←u and repeat the process.

[0165] Theorem 4: For any given value of w and index pair (i,k), Algorithm 3 terminates with an optimal solution to Equation 7. The worst-case time complexity of Algorithm 3 is $O(M^2)$.

[0166] FIG. **8** sets forth Algorithm 4 at element **801**, depicting an example implementation of the SLN-ND functionality for applying a round-robin coordinate descent approach as part of solving an optimization problem, in accordance with aspects of the disclosure.

Coordinate Descent for N-D Optimization:

[0167] The optimization problem defined by Equation 6 is difficult to solve for two reasons, including: (i) the objective function is non-differentiable; and (ii) the optimization problem is non-convex. Therefore, the problem is tackled via a round-robin coordinate descent approach. This is presented in Algorithm 4.

[0168] The following explains the main steps of Algorithm 4. Line 1 performs the initialization of the weights. It then performs coordinate descent until convergence. Lines 4-8 perform normalization so that the N non-negative weights sum up to N. Instead of normalizing after each one-dimensional minimization, processing normalizes it after one round-robin over all N variables. Lines 10-16 perform one round of coordinated descent. For each (i,k), processing performs 1-D minimization of $f(w)$ over w(i,k) by calling Algorithm 3.

[0169] Example 6: Assume the same setting as in Example 5. Let $\mathbb{M}$ contain the 8 matches computed. IoT learning framework **165** may apply Algorithm 4 to this setting, with the initial weights w(1,1)=1, w(2,1)=1, w(2,2)=1. The objective function value is $f(w)=3$, with $\mathbb{M}_w=(\mathbb{M}[1], \mathbb{M}[3], \mathbb{M}[6])$.

[0170] Algorithm 4 performs minimization over w(1,1), there is no improvement. It then performs minimization over w(2,1). The objective function value is reduced to 1 with w(1,1)=1, w(1,1)=1, w(2,1)=2, w(2,2)=1. It then performs minimization over w(2,2), there is no improvement. After normalization, IoT learning framework **165** will have

$$w(1,1)=\frac{3}{4},\ w(2,1)=\frac{3}{2},\ w(2,2)=\frac{3}{4}.$$

The algorithm performs minimization over w(1,1), w(2,1), and w(2,2). None of these produce any improvement. The algorithm terminates, with

$$w(1,1)=\frac{3}{4},\ w(2,1)=\frac{3}{2},\ w(2,2)=\frac{3}{4};\ f(w)=1;$$

$$\mathbb{M}_w=(\mathbb{M}[1], \mathbb{M}[5], \mathbb{M}[8]);\ \text{and}\ \mathbb{S}^w=\left(\mathbb{S}^1(A^1), \mathbb{S}^1(A^2), \mathbb{S}^1(A^2)\right).$$

Proof of Theorem 4:

[0171] Proof of Theorem 4: Algorithm 3 starts with setting w(i,k) to 0, the minimum possible value for w(i,k). It then scans left-to-right to evaluate function values at chosen points.

[0172] Suppose that IoT learning framework **165** is at a current value x≥0 of w(i,k). IoT learning framework **165** may perform the dynamic programming algorithm for weighted interval scheduling to compute a set of matches for the interval scheduling problem of M(1:j) for j=1, 2, . . . , M.

[0173] In addition to computing the current set of intervals, IoT learning framework **165** may also check the trend when w(i,k) is increased by an infinitesimal amount. For this purpose, IoT learning framework **165** may utilize a [j] to denote the number of selected intervals for $\mathbb{M}$M(1:j) whose weight is w(i,k). If IoT learning framework **165** finds $\mathbb{M}[j]\cdot w+OPT[p[j]]>OPT[j-1]$ in Line 8, $\mathbb{M}[j]$ is selected in the optimal solution for $\mathbb{M}(1:j)$. However, if the condition in Line 11 is also true, then the condition in Line 8 will no longer be true if w(i,k) is increased from its current value of x to

$$x+\frac{\mathbb{M}[j]\cdot w + OPT[p[j]] - OPT[j-1]}{a[j-1] - 1[p[j]] - \delta[j]}$$

plus an infinitesimal amount. Therefore, IoT learning framework **165** may add a corresponding upper-bound on σ in Line 12.

[0174] If IoT learning framework **165** finds $\mathbb{M}[j]\cdot w+OPT[p[j]]<OPT[j-1]$ in Line 8, $\mathbb{M}[j]$ is not selected in the optimal solution for $\mathbb{M}(1:j)$, and control goes to Line 14. However, if the condition in Line 16 is also true, then the condition in Line 8 will become true if w(i,k) is increased from its current value of x to

$$x+\frac{\mathbb{M}[j]\cdot w+OPT[p[j]]-OPT[j-1]}{a[j-1]-a[p[j]]-\delta[j]}$$

plus an infinitesimal amount. Therefore, IoT learning framework **165** may add a corresponding upper-bound on σ in Line 17.

[0175] In Lines 19-24, IoT learning framework **165** sets the correct value of a[j] to reflect the trend of change of $\mathbb{M}^w$ when w(i,k) is increased from its current value of x by an infinitesimal amount.

[0176] Algorithm 3 considers all possible cases of the upper bound. If the condition in Line 25 is true, IoT learning framework **165** has reached the right end. Otherwise, IoT learning framework **165** may continue the left-to-right scan. Given the discrete feature of the dynamic programming algorithm, IoT learning framework **165** need not have determined whether the function is left continuous or right continuous at the boundary points. Therefore, IoT learning framework **165** may also evaluate the function value at the mid-point of x and u, which is v. Since the function only takes O(M) different values, the goto statement in Line 36 is executed O(M) times. Since the execution of Line 2 to Line 35 uses O(M) time, the algorithm has a worst-case time complexity O(M2).

[0177] Theorem 5: Algorithm 4 stops after a finite number of iterations. Let w be the weight at the end of the algorithm. Then f(w) cannot be reduced by minimizing over any single variable w(i,k).

[0178] Algorithm 4 does not guarantee finding an optimal solution to Equation 6. Similar to most machine learning algorithms, Algorithm 4 produces a solution to Equation 6 that cannot be improved by optimizing along any of the coordinates. A theoretical bound is not known on the performance gap of the algorithm. However, extensive experiments show that the algorithm performs well. Besides proving that the algorithm converges in a finite number of iterations, there is no determined theoretical bound on the worst-case running time of the algorithm. Given the non-convex nature of the problem, it is unlikely to compute an optimal solution in polynomial time.

Proof of Theorem 5:

[0179] Let $w^{[0]}$ be the initial weight vector. $f(w^{[0]})$ is a non-negative integer. Whenever the algorithm finds an improved solution, the objective function value is reduced by at least 1. Therefore the algorithm is guaranteed to terminate. When the algorithm terminates, it is at a weight vector from which the objective function value cannot be improved by minimization over any of the N variables. This proves the theorem.

Performance Evaluations:

[0180] FIG. **9** depicts a chart of the various performances of E2AP and IoTMosaic on synth/synth-MF test cases, in accordance with aspects of the disclosure. As shown here, each test case consists of a distinct sequence of 2,959 user activities and a corresponding sequence of device events. Each dot in FIG. **9** shows the accuracy achieved by the indicated algorithm on a distinct test case. The chart provides the accuracy of IoTMosaic and E2AP on 200 synthetically generated test cases (100 test cases with device malfunctions, and 100 test cases without device malfunctions) for the distinct sequence of user activities and corresponding sequence of device events, following the distribution of the real data.

[0181] The scheme was implemented and compared it with IoTMosaic. The solution was not compared with or Peek-a-Boo, since many devices used in are simple sensors without Internet connectivity and Peek-a-Boo infers user activities from the states of devices and sensors, rather than solely from the sequence of device events. The evaluation used E2AP to denote the scheme, and IoTMosaic to denote the scheme in. The evaluation studied the performance on both real data from the smart home test-bed, and synthetic data generated following the patterns observed in the real data.

[0182] The evaluation setup and metrics are discussed below, followed by the evaluation results and the observations/analyses. The results show that E2AP exhibits high accuracy and stability, and outperforms IoTMosaic.

[0183] Evaluation Setup and Metrics: The evaluation used the data collected from a smart home, involving 21 distinct user activities and 25 distinct device events from 11 different IoT devices. Experimental data were collected over a period of two months, with a total of 2,959 user activities, and 15,420 corresponding device events (computed from network traffic using IoTAthena). The evaluation calls these data real data and denotes them as real. The evaluations studied three cases: (i) 387 user activities (2,013 events) in the first week, (ii) 1,494 user activities (7,934 events) in the first month, (iii) 2,959 user activities (15,420 events) throughout the experiment.

[0184] In the real data, two device events from two IoT devices (Ring doorbell and Ring spotlight) could be delayed, due to devices on sleep mode. No device malfunctioned during the experiment. The evaluation also used a test case generator, based on the characteristics of the real data collected. In addition to delayed device events, the evaluation added scenarios where up to two devices malfunctioned during part of the time.

[0185] The evaluation calls these data synthetic data, and denote them by synth where no devices malfunctioned, and by synth-MF where up to two devices malfunctioned during part of the time.

[0186] The evaluation ran E2AP and IoTMosaic on both the real data and the synthetic data. Let $\mathbb{E}$ denote the input sequence of device events, let $\mathbb{A}$ denote the input sequence of user activities, let w denote the weight vector computed, let $\mathbb{S}^w$ denote the sequence of activity patterns computed, let $\mathbb{E}^w$ denote the sequence of device events by the concatenation of the elements of $\mathbb{S}^w$, and let $\mathbb{A}^w$ denote the sequence of user activities computed. For IoTMosaic, the experiment measured the number of undetected user activities (denoted by FN), the number of falsely detected user activities (denoted by FP), and the accuracy. For E2AP, the

experiment measured the edit distance between $\mathbb{E}^w$ and $\mathbb{E}$ (denoted by d(E)), the edit distance between $\mathbb{A}^w$ and $\mathbb{A}$ (denoted by d(A)), the number of undetected user activities (denoted by FN), the number of falsely detected user activities (denoted by FP), and the accuracy. Since d(E) and d(A) tend to increase as the lengths of the sequences increase, IoT learning framework **165** may utilize $d_N(E)=d(E)/|\mathbb{E}|$ and $d_N(A)=d(A)/|\mathbb{A}|$ as a form of normalization.

[0187] FIGS. **10**A and **10**B set forth Table 3A at elements **1001**A and **1001**B, depicting results on real data, with one case per row, and further depicting synthetic data, as an average over 100 cases per row, in accordance with aspects of the disclosure.

Evaluation Results and Observations:

[0188] Table 3 (elements **1001**A and **1001**B) provides the evaluation results. For each of the three sizes for real data (first week, first month, two months), the corresponding entries in the table are the results of a single run of the denoted algorithm. For each of the five sizes for synth and synth-MF, the evaluation generated 100 test cases. These 100 test cases have the same number of user activities, but have different sequences of user activities (as these are randomly generated), hence having a different number of device events (as the device events are triggered by different user activities). Each entry for #Event shows the minimum and maximum numbers of events (over 100 test cases). All other entries are the average over 100 test cases. The evaluation observed that E2AP has a higher accuracy than IoTMosaic and exhibits more stability.

[0189] The evaluations show that both IoTMosaic and E2AP are quite accurate. However, E2AP is consistently more accurate than IoTMosaic. For the 100 test cases without device malfunctions, the accuracies of E2AP fall in the interval [99.09%, 99.97%] with a mean of 99.63% and a standard deviation of 0.0016, compared to that of IoTMosaic in the interval [87.28%, 90.78%] with a mean of 89.09% and a standard deviation of 0.0063. For the 100 test cases with device malfunctions, the accuracies of E2AP fall in the interval [97.70%, 99.02%] with a mean of 98.41% and a standard deviation of 0.0030, compared to that of IoTMosaic in the interval [80.11%, 85.31%] with a mean of 82.65% and a standard deviation of 0.0103. The advantage of E2AP over IoTMosaic is more significant over the combined 200 test cases.

[0190] The accuracies of E2AP fall in the interval [97.70%, 99.97%] with a mean of 99.02% and a standard deviation of 0.0066, compared to that of IoTMosaic in the interval [80.11%, 90.78%] with a mean of 85.87% and a standard deviation of 0.0333. The evaluations observe that E2AP is 15% more accurate than IoTMosaic and 5 times more stable than IoTMosaic.

[0191] As described above, IoTMosaic gives higher priorities to exact matches of a signature over partial matches of a signature. When a device malfunctions, events corresponding to this malfunctioned device will not appear in the observed sequence of device events. Therefore, in the presence of malfunctioning devices, a user activity may only trigger a subsequence of its full signature. Therefore the number of exact matches will be reduced. This is the root cause for the lower accuracy and higher instability of IoTMosaic. In contrast, E2AP can intelligently adapt to such situations by learning the proper weights for all possible patterns.

[0192] More experiments were conducted with different synthetic datasets, and found that the results are consistent with those presented in Table 3 (see FIGS. **10**A and **10**B) and the chart provided at FIG. **9**. E2AP is consistently accurate and stable across different experiment scenarios and varying test cases. Conversely, the accuracy of IoTMosaic decreases when the number of missing events increases, which conforms to the analysis above.

[0193] In such a way, the problem of inferring a sequence of user activities together with their patterns from a sequence of device events in a smart home setting is described and resolved. The novel two-phase scheme for solving this problem was specially designed and implemented with the notable extension of the described unsupervised learning algorithm which helps make the inference more adaptive to varying scenarios. No existing algorithm can be directly applied to minimize the non-differentiable and non-convex loss function for the problem.

[0194] Still further, the specially designed novel algorithm for minimizing the loss function over one variable is provided herein, which is a central component in the unsupervised learning algorithm. Extensive evaluations show that the algorithm is significantly more robust and accurate than the state-of-the-art algorithm. Further extensions may include, for instance, evaluating the limitations of the scheme in more smart homes as well as exploring its applications in home security. Another possible extension is to explore other machine learning techniques to solve this problem.

[0195] FIG. **11** is a flow chart illustrating an example mode of operation for computing device **100** to infer user activities from Internet of Things (IoT) connected device events using machine learning based algorithms, in accordance with aspects of the disclosure. The mode of operation is described with respect to computing device **100** and FIGS. **1**, **2**, **3**A, **3**B, **3**C, **4**A, **4**B, **5**, **6**, **7**A, **7**B, **7**C, **8**, **9**, **10**A, and **10**B.

[0196] Computing device **100** may obtain a training dataset (**1105**). For instance, processing circuitry may execute an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model. In such an example, processing circuitry may obtain, using the IoT learning framework, a training dataset indicating at least sequences of IoT device events.

[0197] Computing device **100** may Extract representative user activity patterns (**1110**). For example, processing circuitry **199** of computing device **100** may extract, using the IoT learning framework, representative user activity patterns from the sequences of IoT device events indicated by the training dataset.

[0198] Computing device **100** may train an AI model to learn an optimal subset of the sequences of IoT device events (**1115**). For example, processing circuitry **199** of computing device **100** may train, using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold.

[0199] Computing device **100** may output the ai model (**1120**).

[0200] Computing device **100** may obtain new data indicating new IoT device events (**1125**). For example, processing circuitry **199** of computing device **100** may obtain, using

the IoT learning framework, new data indicating new sequences of IoT device events not indicated by the training dataset.

[0201] Computing device **100** may generate output predicting user activities using the AI model (**1130**). For example, processing circuitry **199** of computing device **100** may generate, using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

[0202] According to another example, computing device **100** may deterministically extract, using the AI model, representative user activity patterns from the sequences of IoT device events indicated by the training dataset. According to such an example, computing device **100** may train, using the IoT learning framework, the AI model using the representative user activity patterns.

[0203] According to another example, computing device **100** may deterministically extract, using the AI model, device events from a plurality of connected IoT devices based on the plurality of connected IoT devices each generating a repeatable sequence of network packets represented within the training dataset. According to such an example, computing device **100** may train, using the IoT learning framework, the AI model using the device events extracted.

[0204] According to another example, computing device **100** may apply, using the AI model, unsupervised learning to the user activity patterns to learn network weights for the IoT learning framework.

[0205] According to another example, computing device **100** may apply, using the IoT learning framework, a loss function to adapt the AI model to device malfunctions indicated within the training dataset.

[0206] According to another example, computing device **100** may generate, using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events satisfying the smallest quantity of the sequences of IoT device events to predict the user activities.

[0207] According to another example, computing device **100** may generate, using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new data satisfying a match for a repeatable sequence of network packets identified within the training dataset.

[0208] For processes, apparatuses, and other examples or illustrations described herein, including in any flowcharts or flow diagrams, certain operations, acts, steps, or events included in any of the techniques described herein may be performed in a different sequence, may be added, merged, or left out altogether (e.g., not all described acts or events are necessary for the practice of the techniques). Moreover, in certain examples, operations, acts, steps, or events may be performed concurrently, e.g., through multi-threaded processing, interrupt processing, or multiple processors, rather than sequentially. Certain operations, acts, steps, or events may be performed automatically even if not specifically identified as being performed automatically. Also, certain operations, acts, steps, or events described as being performed automatically may be alternatively not performed automatically, but rather, such operations, acts, steps, or events may be, in some examples, performed in response to input or another event.

[0209] The detailed description set forth below, in connection with the appended drawings, is intended as a description of various configurations and is not intended to represent the only configurations in which the concepts described herein may be practiced. The detailed description includes specific details for the purpose of providing a thorough understanding of the various concepts. However, it will be apparent to those skilled in the art that these concepts may be practiced without these specific details. In some instances, well-known structures and components are shown in block diagram form in order to avoid obscuring such concepts.

[0210] In accordance with the examples of this disclosure, the term "or" may be interrupted as "and/or" where context does not dictate otherwise. Additionally, while phrases such as "one or more" or "at least one" or the like may have been used in some instances but not others; those instances where such language was not used may be interpreted to have such a meaning implied where context does not dictate otherwise.

[0211] In one or more examples, the functions described may be implemented in hardware, software, firmware, or any combination thereof. If implemented in software, the functions may be stored, as one or more instructions or code, on and/or transmitted over a computer-readable medium and executed by a hardware-based processing unit. Computer-readable media may include computer-readable storage media, which corresponds to a tangible medium such as data storage media, or communication media including any medium that facilitates transfer of a computer program from one place to another (e.g., pursuant to a communication protocol). In this manner, computer-readable media generally may correspond to (1) tangible computer-readable storage media, which is non-transitory or (2) a communication medium such as a signal or carrier wave. Data storage media may be any available media that may be accessed by one or more computers or one or more processors to retrieve instructions, code and/or data structures for implementation of the techniques described in this disclosure. A computer program product may include a computer-readable medium.

[0212] By way of example, and not limitation, such computer-readable storage media may include RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage, or other magnetic storage devices, flash memory, or any other medium that may be used to store desired program code in the form of instructions or data structures and that may be accessed by a computer. Also, any connection is properly termed a computer-readable medium. For example, if instructions are transmitted from a website, server, or other remote source using a coaxial cable, fiber optic cable, twisted pair, digital subscriber line (DSL), or wireless technologies such as infrared, radio, and microwave, the coaxial cable, fiber optic cable, twisted pair, DSL, or wireless technologies such as infrared, radio, and microwave are included in the definition of medium. It should be understood, however, that computer-readable storage media and data storage media do not include connections, carrier waves, signals, or other transient media, but are instead directed to non-transient, tangible storage media. Disk and disc, as used, includes compact disc (CD), laser disc, optical disc, digital versatile disc (DVD), floppy disk and Blu-ray disc, where disks usually reproduce data magnetically, while discs reproduce data optically with lasers. Combinations of the above should also be included within the scope of computer-readable media.

[0213] Instructions may be executed by one or more processors, such as one or more digital signal processors (DSPs), general purpose microprocessors, application specific integrated circuits (ASICs), field programmable logic arrays (FPGAs), or other equivalent integrated or discrete logic circuitry. Accordingly, the terms "processor" or "processing circuitry" as used herein may each refer to any of the foregoing structures or any other structure suitable for implementation of the techniques described. In addition, in some examples, the functionality described may be provided within dedicated hardware and/or software modules. Also, the techniques could be fully implemented in one or more circuits or logic elements.

What is claimed is:

1. A system comprising:

processing circuitry; and

non-transitory computer readable media storing instructions that, when executed by the processing circuitry, configure the processing circuitry to:

execute, by the processing circuitry, an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model;

obtain, by the processing circuitry using the IoT learning framework, a training dataset indicating at least sequences of IoT device events;

extract, by the processing circuitry using the IoT learning framework, representative user activity patterns from the sequences of IoT device events indicated by the training dataset;

train, by the processing circuitry using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold;

output, by the processing circuitry using the IoT learning framework, the AI model;

obtain, by the processing circuitry using the IoT learning framework, new data indicating new sequences of IoT device events not indicated by the training dataset; and

generate, by the processing circuitry using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

2. The system of claim **1**, wherein the processing circuitry is further configured to:

deterministically extract, by the processing circuitry using the AI model, representative user activity patterns from the sequences of IoT device events indicated by the training dataset; and

train, by the processing circuitry using the IoT learning framework, the AI model using the representative user activity patterns.

3. The system of claim **1**, wherein the processing circuitry is further configured to:

deterministically extract, by the processing circuitry using the AI model, device events from a plurality of connected IoT devices based on the plurality of connected IoT devices each generating a repeatable sequence of network packets represented within the training dataset; and

train, by the processing circuitry using the IoT learning framework, the AI model using the device events extracted.

4. The system of claim **1**, wherein the processing circuitry is further configured to:

apply, by the processing circuitry using the AI model, unsupervised learning to the user activity patterns to learn network weights for the IoT learning framework.

5. The system of claim **1**, wherein the processing circuitry is further configured to:

apply, by the processing circuitry using the IoT learning framework, a loss function to adapt the AI model to device malfunctions indicated within the training dataset.

6. The system of claim **1**, wherein the processing circuitry is further configured to:

generate, by the processing circuitry using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events satisfying the smallest quantity of the sequences of IoT device events to predict the user activities.

7. The system of claim **6**, wherein the processing circuitry is further configured to:

generate, by the processing circuitry using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new data satisfying a match for a repeatable sequence of network packets identified within the training dataset.

8. A method comprising:

executing, by one or more processors of a computing device, an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model;

obtaining, by the one or more processors using the IoT learning framework, a training dataset indicating at least sequences of IoT device events;

extracting, by the one or more processors using the IoT learning framework, representative user activity patterns from the sequences of IoT device events indicated by the training dataset;

training, by the one or more processors using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold;

outputting, by the one or more processors using the IoT learning framework, the AI model;

obtaining, by the one or more processors using the IoT learning framework, new data indicating new sequences of IoT device events not indicated by the training dataset; and

generating, by the one or more processors using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

9. The method of claim **8**, further comprising:

deterministically extracting, by the one or more processors using the AI model, representative user activity patterns from the sequences of IoT device events indicated by the training dataset; and

training, by the one or more processors using the IoT learning framework, the AI model using the representative user activity patterns.

10. The method of claim **8**, further comprising:

deterministically extracting, by the one or more processors using the AI model, device events from a plurality of connected IoT devices based on the plurality of connected IoT devices each generating a repeatable sequence of network packets represented within the training dataset; and

training, by the one or more processors using the IoT learning framework, the AI model using the device events extracted.

11. The method of claim **8**, further comprising:

applying, by the one or more processors using the AI model, unsupervised learning to the user activity patterns to learn network weights for the IoT learning framework.

12. The method of claim **8**, further comprising:

applying, by the one or more processors using the IoT learning framework, a loss function to adapt the AI model to device malfunctions indicated within the training dataset.

13. The method of claim **8**, further comprising:

generating, by the one or more processors using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events satisfying the smallest quantity of the sequences of IoT device events to predict the user activities.

14. The method of claim **13**, further comprising:

generating, by the one or more processors using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new data satisfying a match for a repeatable sequence of network packets identified within the training dataset.

15. Computer-readable storage media storing instructions that, when executed, configure processing circuitry to:

execute an Internet-of-Things (IoT) learning framework (IoT learning framework) to train an AI model;

obtain, using the IoT learning framework, a training dataset indicating at least sequences of IoT device events;

extract, using the IoT learning framework, representative user activity patterns from the sequences of IoT device events indicated by the training dataset;

train, using the IoT learning framework, the AI model to learn an optimal subset of the sequences of IoT device events corresponding to a smallest quantity of the sequences of IoT device events to predict user activities with accuracy that satisfies a threshold;

output, using the IoT learning framework, the AI model;

obtain, using the IoT learning framework, new data indicating new sequences of IoT device events not indicated by the training dataset; and

generate, using the AI model, output indicating one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events.

16. The computer-readable storage media comprising of claim **15**, wherein the processing circuitry is further configured to:

deterministically extract, using the AI model, representative user activity patterns from the sequences of IoT device events indicated by the training dataset; and

train, using the IoT learning framework, the AI model using the representative user activity patterns.

17. The computer-readable storage media comprising of claim **15**, wherein the processing circuitry is further configured to:

deterministically extract, using the AI model, device events from a plurality of connected IoT devices based on the plurality of connected IoT devices each generating a repeatable sequence of network packets represented within the training dataset; and

train, using the IoT learning framework, the AI model using the device events extracted.

18. The computer-readable storage media comprising of claim **15**, wherein the processing circuitry is further configured to:

apply, using the AI model, unsupervised learning to the user activity patterns to learn network weights for the IoT learning framework.

19. The computer-readable storage media comprising of claim **15**, wherein the processing circuitry is further configured to:

apply, using the IoT learning framework, a loss function to adapt the AI model to device malfunctions indicated within the training dataset.

20. The computer-readable storage media comprising of claim **15**, wherein the processing circuitry is further configured to:

generate, using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new sequences of IoT device events satisfying the smallest quantity of the sequences of IoT device events to predict the user activities; and

generate, using the AI model, the output indicating the one or more user activities predicted by the AI model to have occurred based on the new data satisfying a match for a repeatable sequence of network packets identified within the training dataset.

* * * * *