(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 January 2022 (13.01.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/010731 A1

(51) International Patent Classification:
G06F 16/28 (2019.01) *G06N 3/08* (2006.01)
G06F 16/25 (2019.01) *G06N 3/04* (2006.01)
G06F 16/22 (2019.01) *G06N 20/00* (2019.01)
G06F 16/2458 (2019.01)

(21) International Application Number:
PCT/US2021/040081

(22) International Filing Date:
01 July 2021 (01.07.2021)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/048,680 07 July 2020 (07.07.2020) US
17/364,125 30 June 2021 (30.06.2021) US

(71) Applicant: **NEC LABORATORIES AMERICA, INC.** [US/US]; 4 Independence Way, Suite 200, Princeton, NJ 08540 (US).

(72) Inventors: **MIZOGUCHI, Takehiko**; 8213 Taylor Court, West Windsor, NJ 08550 (US). **SONG, Dongjin**; 22 Rutgers Lane, Princeton, NJ 08540 (US). **CHEN, Yuncong**; 4912 Fox Run Drive, Plainsboro, NJ 08536 (US). **LUMEZANU, Cristian**; 10 Landing Lane, Princeton Junction, NJ 08550 (US). **CHEN, Haifeng**; 11 Blackhawk Court, West Windsor, NJ 08550 (US).

(74) Agent: **BITETTO, James J.**; Tutunjian & Bitetto, P.C., 401 Broadhollow Road, Suite 402, Melville, NY 11747 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) Title: COMPACT REPRESENTATION AND TIME SERIES SEGMENT RETRIEVAL THROUGH DEEP LEARNING

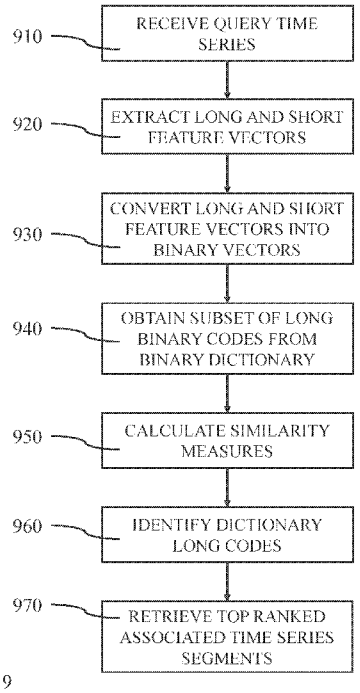

FIG. 9

(57) Abstract: Systems and methods for retrieving similar multivariate time series segments are provided. The systems and methods include extracting (920) a long feature vector and a short feature vector from a time series segment, converting (930) the long feature vector into a long binary code, and converting (930) the short feature vector into a short binary code. The systems and methods further include obtaining (940) a subset of long binary codes from a binary dictionary storing dictionary long codes based on the short binary codes, and calculating (950) similarity measure for each pair of the long feature vector with each dictionary long code. The systems and methods further include identifying (960) a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes, and retrieving (970) a predetermined number of time series segments associated with the predetermined number of dictionary long codes.

WO 2022/010731 A1

[Continued on next page]

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

COMPACT REPRESENTATION AND TIME SERIES SEGMENT RETRIEVAL THROUGH DEEP LEARNING

RELATED APPLICATION INFORMATION

[0001] This application claims priority to Provisional Patent Application No. 63/048,680, filed on July 7, 2020, and U.S. Patent Application No. 17/364,125, filed on June 30, 2021, each incorporated herein by reference in their entirety.

BACKGROUND

Technical Field

[0002] The present invention relates to retrieval of similar multivariate time series and more particularly training and implementation of neural networks for retrieval of similar multivariate time series.

Description of the Related Art

[0003] A time series is a series of observations or data points collected over a period of time that are in time order. A Multivariate time series has more than one time-dependent variable, where values of the variables are collected over a period of time. The data points can be equally spaced in time. Analysis can look for patterns, in particular periodicities in the collected data. Time series forecasting is the use of a model to predict future values based on previously observed values. Multivariate time series data naturally arises in many areas of real-world applications, for example, complex physical systems such as power plants, furnace operations, airplane and automobile engines, and air pollution monitoring. Time series data can arise from monitoring industrial processes or tracking economic and business data. Models for time series data can have many forms and represent different stochastic processes.

[0004] The series of observations or data points collected over the period of time can be identified and stored for later searching and retrieval. To be identified and stored each set of data points can be assigned a binary code as the identifier. The problem of finding optimal binary codes for the data points, however, is NP hard.

SUMMARY

[0005] According to an aspect of the present invention, a computer implemented method of retrieving similar multivariate time series segments is provided. The method includes extracting a long feature vector and a short feature vector from a time series segment, converting the long feature vector into a long binary code, and converting the short feature vector into a short binary code. The method further includes obtaining a subset of long binary codes from a binary dictionary storing dictionary long codes based on the short binary codes, and calculating similarity measure for each pair of the long feature vector with each dictionary long code. The method further includes identifying a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes, and retrieving a predetermined number of time series segments associated with the predetermined number of dictionary long codes.

[0006] According to another aspect of the present invention, a processing system for retrieving similar multivariate time series segments is provided. The system includes one or more processors, and memory coupled to the one or more processors. The system further includes a long feature extractor stored in memory, wherein the long feature extractor is configured to extract a long feature vector from a time series segment, and a short feature extractor stored in memory, wherein the short feature extractor is configured to convert a long feature generated by the long feature extractor

into a shorter length feature through a linear mapping. The system further includes a long binary extractor stored in memory, wherein the long binary extractor is configured to convert a long feature from the long feature extractor into a long binary code having the same length as the long feature, and a short binary extractor stored in memory, wherein the short binary extractor is configured to convert a short feature from the short feature extractor into a short binary code having the same length as the short feature. The system further includes a similarity comparator stored in memory, wherein the similarity comparator is configured to calculate a pairwise similarity between a long binary code extracted from the query and all long binary codes retrieved from a dictionary, and identifying a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes.

[0007] According to yet another aspect of the present invention, a computer program product for retrieving similar multivariate time series segments, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith, the program instructions executable by a computer. The program instructions executable by a computer to cause the computer to perform extracting a long feature vector and a short feature vector from a time series segment, converting the long feature vector into a long binary code, and converting the short feature vector into a short binary code. The program instructions executable by a computer further cause the computer to perform obtaining a subset of long binary codes from a binary dictionary storing dictionary long codes based on the short binary codes, and calculating similarity measure for each pair of the long feature vector with each dictionary long code. The program instructions executable by a computer further cause the computer to perform identifying a predetermined number of dictionary long codes

having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes, and retrieving a predetermined number of time series segments associated with the predetermined number of dictionary long codes.

[0008] These and other features and advantages will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0009] The disclosure will provide details in the following description of preferred embodiments with reference to the following figures wherein:

[0010] FIG. 1 is a block/flow diagram illustrating a high-level system/method for collection of multivariate time series data from a plurality of sensors for subsequent generation of hash codes by a neural network, in accordance with an embodiment of the present invention;

[0011] FIG. 2 is a block/flow diagram illustrating a system/method of a neural network producing and storing a hash code, in accordance with an embodiment of the present invention;

[0012] FIG. 3 is a flow diagram illustrating a system/method for long and short binary code generation using long and short feature extractors, in accordance with an embodiment of the present invention;

[0013] FIG. 4 is a block/flow diagram illustrating an architecture for a system/method of a Deep Sub-Linear Hashing Network (DSLHN), in accordance with an embodiment of the present invention;

[0014] FIG. 5 is a diagram illustrating triplet loss with local boundaries, in accordance with an embodiment of the present invention;

[0015] FIG. 6 is a diagram illustrating triplet loss and cross-entropy with global boundaries, in accordance with an embodiment of the present invention;

[0016] FIG. 7 is a block/flow diagram illustrating a method of training a neural network for hash code generation and retrieval, in accordance with an embodiment of the present invention;

[0017] FIG. 8 is a block/flow diagram illustrating a method of implementing a neural network for hash code generation and retrieval, in accordance with an embodiment of the present invention;

[0018] FIG. 9 is a block/flow diagram illustrating a method of implementing a neural network for time series retrieval, in accordance with an embodiment of the present invention;

[0019] FIG. 10 is an exemplary processing system 1000 to which the present methods and systems may be applied, in accordance with an embodiment of the present invention; and

[0020] FIG. 11 is an exemplary processing system 1000 to which the present methods may be applied to and using LSTM and GRU neural networks and database(s), in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0021] In accordance with embodiments of the present invention, systems and methods are provided for obtaining compact representations of historical time series data for efficient retrieval of the most relevant data from large amounts of historical data through the use of deep learning neural networks. The task of multivariate time

series retrieval can be applied to many tasks in complex systems including system status identification, fault detection and fault prediction. Sensors can also be placed on people for continuously monitoring health status. Large amounts of historical multivariate time series data recorded from a system can be used to understand the current state of the system through comparison to similar failure occurrences. Retrieving relevant multivariate time series segments (i.e., a slice of multivariate time series that lasts for a short time period) from a database by querying with a current time series segment is referred to as multivariate time series retrieval.

[0022] The amount of memory used to identify the stored objects, however, should be small enough so that all object codes can fit in RAM. If this is not the case, i.e. if a significant portion of the object codes have to be stored on a disk, then the response time of a query collapses because the disk access is much slower than that of RAM access. A simpler representation of the time series segments can be obtained to reduce memory usage. Similar samples in raw input space can be mapped to nearby binary codes by minimizing triplet loss, but the boundaries captured by a triplet loss depends on the triplet sample selection. Cross-entropy loss can be used for capturing a global decision boundary.

[0023] In semantic hashing, each item in a database can be represented by a compact binary code. The code is constructed so that similar items will have similar binary codes and a simple feedforward network can calculate the binary code for a new object input.

[0024] In one or more embodiments, a deep neural network can be trained to provide compact binary code representations for input data. Multivariate time series retrieval can then find the most relevant multivariate time series segments from a huge amount of historical data by querying with current observations based on the binary

representations. For example, when a power plant shows some faulty activity, a plant operator may want to refer to similar historical fault cases to identify the specific abnormal status. Therefore, retrieving relevant multivariate time series segments (i.e., a slice of multivariate time series data over a short time period) from a database by querying with the current time series data segment for the present faulty state. Binary codes can preserve in a compact representation the relative similarity relations of the time series historical data in a raw input space. Learning based hashing aims to learn a compact and similarity preserving bit-wise representation such that similar samples in a raw input space are mapped to nearby binary codes by minimizing a triplet loss. Learning based (data dependent) hashing methods build hash function by leveraging the historical training samples.

[0025] In various embodiments, an end-to-end architecture can be provided for efficient multivariate time series retrieval considering a global decision boundary. Only considering relative similarity relation may not capture the global picture of a decision boundary and an expensive step to retrieve multivariate time series data may be needed even with compact binary representation. Even with compact binary representation, an expensive procedure including sorting and a similarity search over all historical data may be needed to retrieve the most relevant time series.

[0026] In various embodiments, Deep Sub-Linear Hashing Network (DSLHN) can be used to perform multivariate time series retrieval and classification. DSLHN employs the Long Short-Term Memory (LSTM) units to extract simple low dimensional features from the input time series segments capturing their temporal dynamics. Two hash functions can predict two different length binary codes, full length binary codes and shorter sub-linear binary codes, from a feature by two serial fully-

connected layers. DSLHN can generate two different length binary codes, full length binary codes and shorter sub-linear ones, from a single time series segment.

[0027] In various embodiments, a classifier is employed to fully utilize the label information in supervised learning-based hashing. A compact binary representation from input data can be data dependent hashing or learning based hashing. Two binary codes of different length can be extracted for each input multivariate time series segment so that efficient similarity searches can be performed.

[0028] In one or more embodiments, deep neural networks including a long feature extractor and a short feature extractor can be utilized to extract segments from entire multivariate time series, and employ recurrent neural network (RNN) such as LSTM/GRU to extract a feature from each segment. A long binary extractor can convert a long feature to a same length binary code by checking the signs of all entries in the feature vector. An extracted long binary code can be stored in a database. The short feature extractor can convert a long feature to a shorter length feature by a linear mapping, and a short binary extractor can convert the short feature to a same length binary code by checking the signs of all entries in the short feature vector. Extracted short binary codes can also be stored in a database. A classifier can compute the probability of belonging to each label and calculates the loss from the misclassification based on the given labels. A sliding window can be used to extract segments from entire multivariate time series, where the length of the sliding window is less than the total length of the time series.

[0029] It is to be understood that aspects of the present invention will be described in terms of a given illustrative architecture; however, other architectures, structures, components and process features and steps can be varied within the scope of aspects of the present invention.

[0030] Referring now in detail to the figures in which like numerals represent the same or similar elements and initially to FIG. 1, a high-level system/method of a block/flow diagram for collection of multivariate time series data from a plurality of sensors for subsequent generation of hash codes by a neural network is illustratively depicted in accordance with an embodiment of the present invention.

[0031] In one or more embodiments, a plurality of sensors 110 can collect sensor readings on a corresponding system being monitored, and output 115 multivariate time series data 120 of the senor readings, where each different sensor A, B, C, D, can produce a different type of time series data. The sensors 110 can be sensors, for example, physical sensors, for measuring, for example, temperature, humidity, vibration, pressure, voltage, current, magnetic field, electrical field, and light sensors, and software sensors, such as logging utilities installed on a computer system to record information regarding the state and behavior of the operating system and applications running on the computer system. The collected multivariate time series data 120 can be composed of a plurality of time series segments 125, 126 that capture particular features of the system behavior from the sensors 110, where the system behavior can be analyzed to discover and/or predict the operation of the system being monitored. The multivariate time series data 120 can be fed 130 into a neural network 140 for analysis and storage, where the neural network 140 can be a deep learning neural network.

[0032] In various embodiments, the neural network 140 can be a recurrent neural network (RNN), for example, a long short term memory (LSTM) or gated recurrent unit (GRU). The neural network can include one or more input nodes 142, hidden nodes 145, and output nodes 147.

[0033] In one or more embodiments, the neural network 140 can include a plurality of neural networks that are trained to produce binary codes from long and short features of the multivariate time series data 120. The neural network 140 can be a deep neural network having one or more hidden layers that include weights for producing the binary codes, where the hidden nodes 145 form the one or more hidden layers, and the hidden layers can be fully connected.

[0034] In various embodiments, a later time series data segment 126 can be the basis for identifying similar earlier time series data segment(s) 125. Time series retrieval tasks aim to identify and retrieve relevant time series from a historical database based on the pair-wise similarity measure between a later query time series segment 126 and the historical time series segments 125.

[0035] In various embodiments, a proposed model employs Long Short-Term Memory (LSTM) units to extract simple low dimensional features from the input time series segments capturing their temporal dynamics. Two different hash functions can predict two different length binary codes from a feature by two serial fully-connected layers. The model can be trained in an end-to-end manner, so that two triplet losses for the two binary codes simultaneously preserve relative similarity measure relations as well as the cross-entropy loss to fully utilize label information for capturing a global decision boundary. Both real value features and their corresponding hash codes can be jointly learned in an end-to-end manner in the neural networks.

[0036] FIG. 2 is a block/flow diagram illustrating a system/method of a neural network producing and storing a hash code, in accordance with an embodiment of the present invention.

[0037] In various embodiments, the neural network 140 can be trained to generate and output 150 a separate hash code 160 for each segment 125, 126 of the multivariate

time series data 120, where the neural network 140 can be trained to generate a short hash code to provide a smaller search space with an increased searching efficiency, and/or a long hash code that is the same length as a long feature. The long hash code and short hash code can be stored 170 in a database 180 for subsequent searching and retrieval.

[0038] In various embodiments, two different length binary codes (hash codes) 160 enable sub-linear searching, which involves searching only a subset of the historical time series data, as specified by the sub-linear binary codes. The binary codes can also map images that are similar (either in terms of feature space distance or semantic distance) to binary strings with a low Hamming distance.

[0039] In various embodiments, a deep neural network can learn the parameters of the network by using three criterions for the codes obtained at the top layer of the network: 1) minimizing loss between the original real-valued feature and the learned binary vector; 2) binary codes distribute evenly on each bit, and 3) each bit is as independent as possible. The parameters of the neural networks can be updated by back-propagation based on the optimization objective function at the top layer.

[0040] In various embodiments, two triplet losses can be employed for these two binary codes to simultaneously preserve relative similarity measure relations. A cross-entropy loss can be used to fully utilize label information for capturing the global decision boundary in the latent space. The Deep Sub-Linear Hashing Network (DSLHN) can be trained in end-to-end manner by minimizing the sum of the two triplet losses and the cross-entropy loss with backpropagation over an entire network based on stochastic gradient descent. A sub-linear search that requires searching only a subset of historical data specified by sub-linear binary codes can then be performed for a query.

[0041] FIG. 3 is a block/flow diagram illustrating a system/method for long and short binary code generation using long and short feature extractors, in accordance with an embodiment of the present invention.

[0042] In various embodiments, a multivariate time series 120 including multiple time series segments can be fed into a long feature extractor 310 that can utilize a sliding window to extract the segments 125 from the entire multivariate time series 120. A recurrent neural network (RNN), for example, an LSTM or GRU, can be used to extract a long feature from each segment 125, 126, where each segment can be a slice of the multivariate time series that lasts for a predefine number of time steps (e.g., a duration or time period). The time series segment 126 can be the most recent time series segment from the time series 120.

[0043] In various embodiments, a long binary extractor 320 can receive and convert a long feature from the long feature extractor 310 into a long binary code 330 having the same length as the long feature. The long binary code 330 can be stored in a database.

[0044] In various embodiments, a short feature extractor 340, which can be a recurrent neural network (RNN), for example, an LSTM or GRU, can be used to convert a long feature generated by the long feature extractor 310 into a shorter length feature through a linear mapping.

[0045] In various embodiments, a short binary extractor 350 can receive and convert a short feature from the short feature extractor 340 into a short binary code 360 having the same length as the short feature by checking the sign of the entries in a short feature vector. The short binary code 360 can be stored in a database. In various embodiments, the short binary code 360 is much shorter than the long binary code 330, where, for example, a long code can be 256 bits long and the short code can be 32 bits long. A

short code can be, for example, $1/8^{th}$ the length of the long code, or the short code can be about $1/4^{th}$ to about $1/16^{th}$ the length of the long code, or the short code can be about $1/6^{th}$ to about $1/12^{th}$ the length of the long code, althogh other length relationships are also contemplated.

[0046] In various embodiments, a classifier 370 can receive a short feature and compute the probability of the short feature belonging to a class, where each class is identified as a label 380, and calculate a loss from misclassification by the classifier based on the provided label(s) 380. The losses can be used to update the parameters of the long feature extractor 310 and/or short feature extractor 340. The losses can be triplet losses for both the long and short binary codes, as well as cross-entropy losses for short features. The parameters can be updated based on triplet minimization. In various embodiments, a classifier 370 is multi-class classifier including different classes from the ground truth provided by label(s) 380, that can compute the probability of the short feature belonging to each class identified by the labels. For example, if there are three classes c1,c2 and c3, classifier 370 calculates a probability of a short feature "f" belonging to each class, i.e., it calculates p(c1|f), p(c2|f) and p(c3|f).

[0047] In various embodiments, after training is finished, the hashing process can be conducted using a new time series segment. A long binary dictionary can be constructed that stores the set of long binary codes that have the same bit pattern as a short binary code.

[0048] In various embodiments, a labeled multivariate time series segment ($\mathbf{X}$, y), where y denotes the label, is denoted as a tuple of d-dimensional and w-length segment $\mathbf{X} = [x^1, x^2, ..., x^d]^T = [x_1, x_2, ..., x_w] \in \mathbb{R}^{d \times w}$ and the label $y \in C$, where w is the length of the window, $x^k = [x^k_1, x^k_2, ..., x^k_w] \in \mathbb{R}^w$ (k = 1, 2, ..., d) is a time series segment of

length w, $x_t = [x^1_t, x^1_t, ..., x^d_t] \in \mathbb{R}^w$ (t = 1, 2, ..., w) is a vector of from all dimensions of the time series segment at a certain time point t, and C is the set of all class labels.

[0049] Suppose there is a collection of historical time series segments denoted by $\mathcal{D} = \{X_i\}_{i=1}^N$, where N is the total number of segments in the collection. Given a newly incoming multivariate time series segment query $X_q \notin \mathcal{D}$, i.e., a slice of d-dimensional time series which lasts w time steps that was not previously a component of the set (e.g., time series segment 126), the time series retrieval task is to find the time series segments in $\mathcal{D}$ most similar to the new time series segment 126, i.e., that is to obtain:

[0050] $\mathcal{X}_q^* \in arg\ min\ \mathcal{S}(X_q, X_p), X_p \in \mathcal{D}$;

[0051] where p is the index of p^{th} segment (p $\in$ {1, 2, ... , N}) for N segments, and $\mathcal{S} : \mathbb{R}^{d \times w} \times \mathbb{R}^{d \times w} \rightarrow [0, \infty)$ is a function which measures the similarity between two multivariate time series segments. This can be utilized for calculating the similarity measure for each pair of the long feature vectors with each of a dictionary long code(s).

[0052] Feature Extraction Layer 410: To perform multivariate time series retrieval efficiently, a good, simple representation of raw multivariate time series segments capturing their temporal dynamics are obtained. In the feature extraction layer 410, given a multivariate time series segment $X = [x_1, x_2, ..., x_d] \in \mathbb{R}^{d \times w}$, where $x_t \in \mathbb{R}^d$ ($1 \leq t \leq d$), we learn a non-linear feature extraction function $f : \mathbb{R}^{d \times w} \rightarrow \mathbb{R}^m$ from **X** to a simple m-dimensional ($m << d \times w$) representation (feature) h, where $h \in \mathbb{R}^m$ with $h := F(\mathbf{X})$. In various embodiments, for example, m = 256, and $d \times w > 20{,}000$, where $d \times w$ can be in a range of about 50 times (50 x) to about 100 times (100 x), or about 75 times (50 x) to about 80 times the value of m.

[0053] In various embodiments, to extract features from multivariate time series segments, a LSTM can be utilized as F, since an LSTM is simple, explicitly captures

both the temporal dynamics and the long-term dependencies of the inputs, and can be used for sequence to sequence learning. Each LSTM unit is composed of a memory cell with the state $\mathbf{s}_t$ and three sigmoid gates: the forget gate $\mathbf{f}_t$, input gate $\mathbf{i}_t$ and output gate $\mathbf{o}_t$ ($\mathbf{s}_t$, $\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t \in \mathbb{R}^m$), which control the access to the memory cell. The update of an LSTM unit can be summarized as:

[0054] $\mathbf{f}_t := \sigma(\mathbf{W}_f [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_f)$,

[0055] $\mathbf{i}_t := \sigma(\mathbf{W}_i [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_i)$,

[0056] $\mathbf{o}_t := \sigma(\mathbf{W}_o [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_o)$,

[0057] $\mathbf{s}_t := \mathbf{f}_t \odot \mathbf{s}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_s [\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_s)$,

[0058] $\mathbf{h}_t := \mathbf{o}_t \odot \tanh(\mathbf{s}_t)$,

[0059] where $[\mathbf{h}_{t-1}; \mathbf{x}_t] \in \mathbb{R}^{m+d}$ is the vertical concatenation of the previous hidden state h_{t-1} and the current input $\mathbf{x}_t$, $\sigma : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is an element-wise logistic sigmoid function and $\odot$ is an element-wise multiplication operator (i.e., Hadamard product).

[0060] Weights $\mathbf{W}_f$; $\mathbf{W}_i$; $\mathbf{W}_o$; $\mathbf{W}_s \in \mathbb{R}^{m \times (m+d)}$ and biases b_f, b_i, b_o, $b_s \in \mathbb{R}^{m \times (m+d)}$ are the parameters to be learned, where the weights can be in the form of matrices. In the feature extractor, the last hidden state of LSTM units h_w is employed as the feature (simple representation) of a raw multivariate time series segment because it encodes temporal dynamic information from the entire segment.

[0061] FIG. 4 is a block/flow diagram illustrating an architecture for a system/method of a Deep Sub-Linear Hashing Network (DSLHN), in accordance with an embodiment of the present invention.

[0062] Feature-Binary Layer 420: Even with a simple representation of a multivariate time series, to retrieve historical time series using a query is a time consuming process involving calculating the similarity of all pairs between the query and the historical data, and sorting the pairs based on their similarity. To avoid this

process, a sub-linear search strategy can be employed, which utilizes much simpler binary representation for efficient multivariate time series retrieval.

[0063] In various embodiments, in a feature-binary layer 420, two kinds of binary codes 330, 360 with different lengths, v_1-bit full-length binary codes and v_2-bit sub-linear binary codes, with the length of v_1 greater than v_2, ($v_1 > v_2$), can be extracted from the output of the feature extraction layer 410, which can include the long feature extractor 310 and the short feature extractor 340, which can be implemented as recurrent neural networks (RNNs).

[0064] Binary code prediction functions: Given the representation for a raw multivariate time series segment h_w, we aim to learn two mappings $H_1 : \mathbb{R}^m \rightarrow \{-1, +1\}^{v1}$ and $H_2 : \mathbb{R}^m \rightarrow \{-1, +1\}^{v1}$, which each compress an m-dimensional real-valued input **h** into respectively v1-bit and v2-bit binary codes. These mappings are referred to as whole binary embedding or hash functions in the literature and are expressed as:

[0065] $H_i(\mathbf{h}) = \text{sgn}(G_i(\mathbf{h}))$, $i = 1, 2$,

[0066] where sgn(·) is the element-wise sign function that extracts the sign of each element in the input, and $G_i : \mathbb{R}^m \rightarrow \mathbb{R}^{vi}$ ($i = 1, 2$) is a prediction function represented by FC1 and FC2. H_1 and H_2 are each hash functions. A variety of prediction function are available for serving to specific data domains and practical applications. In various embodiments, linear prediction functions for G_1 and G_2, i.e.:

[0067] $G1(h) := \mathbf{W}_1\mathbf{h} + b_1$,

[0068] $G2(h) := \mathbf{W}_1 G1(\mathbf{h}) + b_2$,

[0069] where $Wi \in \mathbb{R}^{vi \times m)}$ ($i = 1, 2$) is a weight matrix to be learned. To make each bit nearly balanced and thus take as much information as possible, the bias terms $b_1 = -W_1\bar{h}, and\ b_2 = -W_2\bar{g}$, respectively with the means of **h** and $G_1(h)$ over all samples,

[0070] $\bar{h} = \sum_{i=1}^{N} F(X_i)$, and

[0071] $\bar{g} = \sum_{i=1}^{N} G_1(F(X_i))$.

[0072] The whole hash functions H_1 and H_2 can be:

[0073] $H_1(\mathbf{h}; \mathbf{W}_1) := \mathrm{sgn}(\mathbf{W}_1(\mathrm{h} - \bar{h}))$,

[0074] $H_2(\mathbf{h}; \mathbf{W}_2) := \mathrm{sgn}(\mathbf{W}_2(\mathrm{G}_1(\mathrm{h}) - \bar{g}))$,

[0075] which are parameterized respectively by W_1 and W_2. In the following description, we simply use $H_1(h)$ and $H_2(h)$ for denoting $H_1(\mathbf{h}; \mathbf{W}_1)$ and $H_2(\mathbf{h}; \mathbf{W}_2)$, respectively.

[0076] Triplet losses: Desired hash functions should keep relative similarity relationships in output (Hamming) space between two binary codes from that between two multivariate time series in input space. Rather than considering only pair-wise similarities, relative similarities in the form of triplets $(a, p, n) \in T_{\text{triplet}}$, can be leveraged, whose indices pair (a, p) specifies more similar input segment pair (X_a, X_p) than the segment pair (X_a, X_n) assigned by (a, n), where "a" refers to anchor, "n" refers to negative, and "p" refers to positive. The triplet loss (e.g., anchor, positive, negative) can be employed to ensure that a Hamming distance between an anchor and a positive is less than a Hamming distance between the anchor and a negative. T_{triplet}, is the set { } of all possible triplet indices. The triplets are selected based on class labels, e.g., (a, p, n), which are selected so that Xa and Xp belong to the same class, while X_a and X_n belong to different classes. Intuitively, the desired hash functions $H_i(\cdot)$ (i = 1; 2) would be expected to preserve these relative similarity relationships revealed by L_{triplet}, within the Hamming space, i.e., to make the Hamming distance between the embeddings $H_i(h_a)$ and $H_i(h_p)$ smaller than that between $H_i(h_a)$ and $H_i(h_n)$, where h_a, h_p and h_n are respectively anchor, positive, and negative features extracted from X_a, X_p and X_n by F $(\cdot)$.

[0077] The triplet losses that evaluate hash functions H_i (i = 1; 2) under the above intuition are then:

[0078] $\ell_i^{triplet} := \sum_{(a,p,n)\in\mathcal{L}_{triplet}} max(0, d_i^p - d_i^n + \alpha),\ (i = 1, 2),$

[0079] where $d_i^q := \left\|H_i(h_a) - H_i(h_q)\right\|_0$ is the Hamming distance between $H_i(h_a)$ and $H_i(h_q)$ ($q \in \{p, n\}$), $\|h\|_0$ is the ℓ_0-norm, which counts the number of non-zero entries in h, and $\alpha \geq 0$ is a margin.

[0080] That equation just defines d_i^p and d_i^n, q can be either p or n. h_q is either h_p or h_n discussed above.

[0081] Classification Layer 430: The triplet losses are sufficiently powerful if features or binary codes have rich information that can capture the dynamics of inputted multivariate time series. However, triplet losses are still based on a relative distance relationship, and thus does not consider the global picture of a decision boundary in the feature space. This may have a large influence if the decision boundaries in the Hamming space are obscure, since short binaries like sub-linear binary codes by $H_2(\cdot)$ have poor information to fully represent input time series segments. Therefore, information from class labels can be fully utilized to differentiate feature representation around the boundaries if they are available.

[0082] The classification layer 430 can contain a fully connected (FC) network 370, FC3, that computes the logits 440, $z = \mathbf{W}_c\mathbf{u}$, where $\mathbf{W}_c \in \mathbb{R}^{|C|\times v2}$ is the weight matrix to be learned. $\mathbf{u} := G_2(\mathbf{h})$ for the sub-linear feature extracted by $G_2(\ )$. FC1 and FC2 can compresses (reshape) the intermediate features to desired dimensional features, for example, FC1 reshapes the LSTM output to v1 dimensional features by $G1\ (h) := \mathbf{W}_1\mathbf{h} + b_1$.

[0083] In various embodiments, a softmax layer is added to compute the predicted probability by $\hat{y} := \exp(z)/\sum_{j=1}^{|C|} exp(z_j) \in [0, 1]^{|C|}$, where z_j is the j-th entry in z.

[0084] Cross-Entropy Loss: To provide differentiated feature representations between different classes, following the standard classification strategy, a cross-entropy loss can be utilized for penalizing misclassifications in the sub-linear feature space.

[0085] $\ell^{ce} := \sum_{i=1}^{N} \mathbb{I}_{yi}^{\intercal} \log(\hat{y}_i) + \left(\mathbb{I} - \mathbb{I}_{yi}\right)^{\intercal} \log(\mathbb{I} - \hat{y}_i)$,

[0086] where $\mathbb{I}_{yi} \in \{0, 1\}^{|C|}$ is the one-hot representation of yi and $\hat{y}_i$ is the predicted probability both for the input time series segment $\mathbf{X}_i \in D$, and $\mathbb{I}$ is the $|C|$-length vector of all ones.

[0087] All loss functions can be summarized as the following:

[0088] $\ell(\theta) := \sum_{i=1}^{2} \ell_i^{triplet}(\theta) + \lambda_{ce}\ell^{ce}(\theta)$,

[0089] where θ is the set of all trainable parameters in the model, i.e., $\theta := \{W_f, W_i, W_o, W_s, b_f, b_i, b_o, b_s, W_1, W_2, W_c\}$, and $\lambda_{ce} \geq 0$ is the weight parameter that controls the importance of the cross-entropy loss, ℓ^{ce}.

[0090] Unfortunately, the objective is hard to optimize because the hash functions $H_i(\cdot)$ (i = 1; 2) are discrete mappings and the Hamming distances in the triplet losses $\ell_i^{triplet}$ (i = 1; 2) lie in discrete spaces. Therefore, the network architecture is discrete in nature and its associated optimization problem is combinatorially difficult. To address this issues, the original discrete objective can be relaxed to a continuous and differentiable one. The hash functions $H_1(\cdot)$ and $H_2(\cdot)$ can be relaxed as:

[0091] $H_1(h) \approx \bar{H}_1(h) := tanh\left(W_1\left(h - \bar{h}\right)\right)$,

[0092] $H_2(h) \approx \bar{H}_2(h) := tanh\left(W_2\left(G_1(h) - \bar{g}\right)\right)$,

[0093] which are differentiable. This relaxation is based on the standard approximation sgn(·) ≈ tanh(·). The Hamming distance can also be relaxed to the ℓ_1-distance, i.e., $d_i^q \approx \bar{d}_i^q := \left\|\bar{H}_i(h_a) - \bar{H}_i(h_q)\right\|_1$ (q ∈ {n, p}, i = 1; 2).

[0094] $\bar{H}_i$ is either $\bar{H}_1$ or $\bar{H}_2$.

[0095] Based on the above relaxations, we finally have the following continuous and differentiable objective:

[0096] $\bar{\ell}(\theta) := \sum_{i=1}^{2} \bar{\ell}_i^{triplet}(\theta) + \lambda_{ce}\ell^{ce}(\theta)$,

[0097] where

[0098] $\bar{\ell}_i^{triplet} := \sum_{(a,p,n)\in\mathcal{T}_{triplet}} max(0, \bar{d}_i^p - \bar{d}_i^n + \alpha),\ (i = 1, 2)$.

[0099] These relaxations have been naturally used for the optimization of binary embedding networks. An Adam optimizer can be employed to perform backpropagation over the entire network based on stochastic gradient descent (SGD) with a mini-batch size of 256 for optimizing the trainable parameters θ of the proposed network.

[0100] Multivariate Time Series Retrieval Based on Sub-linear Search:

[0101] If the training is finished, two different length of binary codes, $c_i^{full} \in \{0, 1\}^{v1}$ and $c_i^{sub} \in \{0, 1\}^{v2}$ can be extracted for the historical time series segments $X_i \in D$ respectively by $c_i^{full} := H_1(F(X_i))$ and $c_i^{sub} := H_2(F(X_i))$.

[0102] Since $v_2 < v_1$, the number of unique sub-linear binary codes, c_i^{sub}, extracted from $\mathbf{X}_i$ (i = 1, ..., N) are expected to be much less than that of unique full-length binary codes, c_i^{full}, i.e., many different full-length binary codes are expected to share the same sublinear binary code. This fact enables us to perform efficient multivariate time series retrieval by a sub-linear search.

[0103] The sub-linear search algorithm for efficient multivariate

[0104] time series retrieval is summarized in Algorithm 1:

[0105] **Input** : $\mathbf{X}_q$, L, k, r_{max}

[0106] **Output** : Top-k similar time series segments to X_q

[0107] 1) $\mathcal{J} \leftarrow \emptyset, r \leftarrow \emptyset$;

[0108] 2) $c_q^{full} \leftarrow H_1\left(F\left(X_q\right)\right), c_q^{sub} \leftarrow H_2\left(F\left(X_q\right)\right)$;

[0109] 3) **while** $|\mathcal{J}| < k$ ***and*** $r < r_{max}$ **do**

[0110] 4) $\Omega_r \leftarrow \left\{c \in \{0,1\}^{v_2} | \left\|c - c_q^{sub}\right\|_0 = r\right\}$;

[0111] 5) **for** $c' \in \Omega_r$ **do**

[0112] 6) $\hookrightarrow \mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{L}(c')$;

[0113] 7) $r \leftarrow r + 1$;

[0114] 8) $\Delta \leftarrow \left\{\left\|c_q^{full} - c_q^{sub}\right\|_0 | j \in \mathcal{J}\right\}$;

[0115] 9) $[i_1^*, i_{21}^*, \ldots, i_{k1}^*] \leftarrow argsort(\Delta)[:k]$;

[0116] 10) **return** $X_{i_1^*}, \ldots, X_{i_k^*}$.

[0117] After extracting full-length and sub-linear binary codes for all historical time series segments, we construct a dictionary L which returns the set of all indices that have the same sub-linear binary code, i.e.,

[0118] Note that the items in L are disjoint,

[0119] $\mathcal{L}(c^{sub}) := \left\{i \middle| c_i^{sub} = c^{sub}\right\} \subset \{1, \ldots, N\}$.

[0120] Note that the items in L are disjoint, i.e., $\mathcal{L}(c_i) \cap \mathcal{L}(c_i) = \emptyset$ if $c_i \neq c_j$.

[0121] For a query time series segment X_q, the full-length and sub-linear binary codes, c_q^{full} and c_q^{sub} , are extracted by DSLHN (line 2). Then, we first retrieve indices of time series segment in database by $\mathcal{L}\left(c_q^{sub}\right)$ and add them to the candidate indices J (lines 4-6 for r = 0). If we do not retrieve sufficient number of indices at this time, i.e., $|\mathcal{J}| < k$, we next look for L with the second nearest sub-linear binary codes, i.e., Ω_r,

sub-linear binary codes, r(≥ 1) of whose bit(s) is (are) different from c_q^{sub} . We iterate this process incrementing r until we have enough candidates (i.e., $|\mathcal{J}| \geq k$) up to the pre-defined maximum number of flipped bits r_{max} (lines 3-7).

[0122] Once we have enough numbers of candidate indices, we calculate pair-wise Hamming distances Δ between full-length binary code of the query segment c_q^{full} and those of the subset of database segments assigned by *J* (line 8). Then, we sort Δ in ascending order and retrieve up to k number of indices from the top ones (line 9), for example, we retrieve j' as i_1* if $\left\|c_q^{full} - c_q^{sub}\right\|_0$ is the smallest within Δ. Finally, we retrieve time series segments $Xi_1; : : : ; Xi_k$.

[0123] Complexity: Here, we discuss the time complexity of the sub-linear search algorithm. This algorithm mainly separated into dictionary access (lines 4-6) and re-ranking (line 9). For general r_{max}, the number of dictionary accesses could be exponential w.r.t. r_{max}, i.e., $\mathcal{O}\left(\binom{v_2}{r}\right) = \mathcal{O}v_2^r$ in the worse case scenario, so we fix r_{max} to at most 2 in practice to avoid combinatorial explosion. For the re-ranking part, it has $|\mathcal{J}|\mathcal{O}(|\mathcal{J}|\log|\mathcal{J}|)$ time complexity. The number of candidate $|\mathcal{J}|$ depends on the properties of the set of historical time series segments. It would be k in the best case while N is the worst case scenario, but no more than the full linear search complexity O(N log N).

[0124] The hyper-parameter λ_{ce} of DSLHN is optimized based on grid search over λ_{ce} 2 f0:001; 0:01; 0:1; 1:0g.

[0125] Most of the variation in x can be accounted for by *m* principal components (PCs), where *m* « p (the total number of variable components). A reduction in complexity and corresponding reduction in data size is achieved by transforming the original variables to the principal components and the corresponding reduction in the

number of variables storing the information. Transforming high-dimensional real-valued object descriptors into compact binary codes can address both memory usage and computational problems. the transformation and resulting compression of data enables storage of a large number of binary codes in memory. A small Hamming distance between codes for similar objects allows queries to be limited to a neighborhood around the binary code associated with the searched feature; thereby further reducing query time and processor usage. The objects with codes within a small Hamming distance of the code for the query can then be retrieved. The Hamming distance can be efficiently computed with hardware (i.e., CPUs, multi-core graphics processors), which can compute millions of distances per second.

[0126] For all cases, deep learning based methods LSTM+triplet, DSLHN(w/o CE) (CE = cross-entropy) and DSLHN consistently outperform shallow methods LSH and ITQ because deep leaning based approach can capture temporal dynamics within time series segments. Within deep learning based methods, our proposed DSLHN provides the best performance in almost all cases for both retrieval and classification tasks. We also find that the proposed DSLHN constantly outperforms DSLHN(w/o CE). It implies cross-entropy loss surely improves both retrieval and classification performance in our model.

[0127] FIG. 5 is a diagram illustrating triplet loss with local boundaries, in accordance with an embodiment of the present invention.

[0128] This may have a large influence if the decision boundaries in the Hamming space are obscure, since short binaries like sub-linear binary codes by $H_2(\cdot)$ have poor information to fully represent input time series segments. The sublinear hashcode pattern 510 does not uniquely map to the individual classes 520. A subset 512 of the hashcodes may map to two or more classes, due to local minima. Therefore,

information from class labels can be fully utilized to differentiate feature representation around the boundaries if they are available. However, considering local boundaries may not be sufficient if there is only poor information like sub-linear binary codes.

[0129] FIG. 6 is a diagram illustrating triplet loss and cross-entropy with global boundaries, in accordance with an embodiment of the present invention.

[0130] The addition of a cross-entropy loss can further differentiate features based on global minima, so each sublinear hash code maps to a single class 520. The subset 512 of sublinear hashcodes that mapped to two or more classes without the cross-entropy loss can thereby be eliminated even though two or more different hashcodes 510 may map to the same class 520.

[0131] FIG. 7 is a block/flow diagram illustrating a method of training a neural network for hash code generation and retrieval, in accordance with an embodiment of the present invention.

[0132] At block 710, a slice of the multivariate time series that lasts for a predetermined number of time steps is extracted from the entire multivariate time series 120 using a sliding window. The length of sliding window can depend on how the time series data is collected, for example, if the data is recorded every minutes for five consecutive days, a sliding window of length 60 can be used for summarizing observations in a 1 hour time window.

[0133] At block 720, long and short feature vectors are extracted utilizing a recurrent neural network.

[0134] At block 730, binary vectors are generated from the long and short feature vectors by checking the signs of all entries of features.

[0135] At block 740, triplet losses are calculated for both long and short binary codes.

[0136] At block 750, a cross-entropy loss is calculated for the short binary codes to differentiate feature representation between different classes.

[0137] At block 760, the parameters of the neural network(s) are updated based on the triplet losses and cross-entropy loss.

[0138] FIG. 8 is a block/flow diagram illustrating a method of implementing a neural network for hash code generation and retrieval, in accordance with an embodiment of the present invention.

[0139] After training has been completed, a hashing process can be conducted.

[0140] At block 810, a slice of the multivariate time series that lasts for a predetermined number of time steps is extracted from the entire multivariate time series 120 using a sliding window. This can be a new, and yet unseen, time series segment that was not used for training or validation of the neural network(s).

[0141] At block 820, long and short feature vectors are extracted for the time series segments utilizing the trained recurrent neural network(s).

[0142] At block 830, binary vectors are generated from the long and short feature vectors generated by the trained neural networks by checking the signs of all entries of features.

[0143] At block 840, long and short binary codes are stored in a database.

[0144] At block 850, a binary dictionary that stores the set of long binary codes that have the same bit pattern as the short binary code(s) can be constructed.

[0145] FIG. 9 is a block/flow diagram illustrating a method of implementing a neural network for time series retrieval, in accordance with an embodiment of the present invention.

[0146] At block 910, the system can receive a time series segment for a query and retrieval of similar time series segments.

[0147] At block 920, for the current observed time series segment, long and short features are extracted based on the recurrent neural network learned during the training.

[0148] At block 930, long and short feature vectors of the query time series segment are converted to long and short binary codes, respectively, by checking signs of all entries in those feature vectors.

[0149] At block 940, the subset of long binary codes that have the same short binary code as extracted from the long and short feature vectors of the query time series segment are retrieved from the binary dictionary constructed in the hashing stage. A sufficient number of long binary codes should be obtained from the dictionary, where a sufficient number is a value larger than k to retrieve top-k similar samples from the database.

[0150] At block 950, a pairwise similarity can be calculated between a long binary code extracted from the query and all long binary codes retrieved from the dictionary.

[0151] At block 960, a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes are identified.

[0152] At block 970, based on the calculated similarities, retrieve a predetermined number of multivariate time series segment identified as the most relevant to the query. The retrieved number of multivariate time series segment can be used for generating an output including a visual representation of the relevant time series segment(s) on a user interface, for example, a display or mobile user device. The predetermined number of time series segments can be displayed to one or more users, where the displayed time series segment(s) can indicate a condition or status of the monitored system to the user. The predetermined number is how many samples we want to see from the most relevant.

[0153] FIG. 10 is an exemplary processing system 1000 to which the present methods and systems may be applied, in accordance with an embodiment of the present invention.

[0154] The processing system 1000 can include at least one processor (CPU) 1004 and may have a graphics processing (GPU) 1005 that can perform vector calculations/manipulations operatively coupled to other components via a system bus 1002. A cache 1006, a Read Only Memory (ROM) 1008, a Random Access Memory (RAM) 1010, an input/output (I/O) adapter 1020, a sound adapter 1030, a network adapter 1040, a user interface adapter 1050, and/or a display adapter 1060, can also be operatively coupled to the system bus 1002.

[0155] A first storage device 1022 and a second storage device 1024 are operatively coupled to system bus 1002 by the I/O adapter 1020, where a neural network can be stored for implementing the features described herein. The storage devices 1022 and 1024 can be any of a disk storage device (e.g., a magnetic or optical disk storage device), a solid state storage device, a magnetic storage device, and so forth. The storage devices 1022 and 1024 can be the same type of storage device or different types of storage devices.

[0156] A speaker 1032 can be operatively coupled to the system bus 1002 by the sound adapter 1030. A transceiver 1042 can be operatively coupled to the system bus 1002 by the network adapter 1040. A display device 1062 can be operatively coupled to the system bus 1002 by display adapter 1060.

[0157] A first user input device 1052, a second user input device 1054, and a third user input device 1056 can be operatively coupled to the system bus 1002 by the user interface adapter 1050. The user input devices 1052, 1054, and 1056 can be any of a keyboard, a mouse, a keypad, an image capture device, a motion sensing device, a

microphone, a device incorporating the functionality of at least two of the preceding devices, and so forth. Of course, other types of input devices can also be used, while maintaining the spirit of the present principles. The user input devices 1052, 1054, and 1056 can be the same type of user input device or different types of user input devices. The user input devices 1052, 1054, and 1056 can be used to input and output information to and from the processing system 1000.

[0158] In various embodiments, the processing system 1000 may also include other elements (not shown), as readily contemplated by one of skill in the art, as well as omit certain elements. For example, various other input devices and/or output devices can be included in processing system 1000, depending upon the particular implementation of the same, as readily understood by one of ordinary skill in the art. For example, various types of wireless and/or wired input and/or output devices can be used. Moreover, additional processors, controllers, memories, and so forth, in various configurations can also be utilized as readily appreciated by one of ordinary skill in the art. These and other variations of the processing system 1000 are readily contemplated by one of ordinary skill in the art given the teachings of the present principles provided herein.

[0159] Moreover, it is to be appreciated that processing system 1000 is a system for implementing respective embodiments of the present methods/systems. Part or all of processing system 1000 may be implemented in one or more of the elements of FIGs. 1-8. Further, it is to be appreciated that processing system 1000 may perform at least part of the methods described herein including, for example, at least part of the method of FIGs. 1-8.

[0160] FIG. 11 is an exemplary processing system 1000 to which the present methods may be applied to and using LSTM and GRU neural networks and database(s), in accordance with an embodiment of the present invention.

[0161] In various embodiments, the neural network (s) (e.g., LSTMs, GRUs, etc.) can be implemented on the processing system 1000, where the long short term memories 1140 of the feature extractors and GRUs of the similarity comparators 1130 may be stored in storage device 1024. The similarity comparator 1130 stored in memory can be configured to calculate a pairwise similarity measure between a long binary code extracted from the query and all long binary codes retrieved from a dictionary, and identifying a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes. The received and collected time series data 120 can be stored in a database that may reside in the first storage device 1022 and/or the second storage device 1024. The sensors 110 can be connected to and in electronic communication with system 1000 through network adapter 1040 and/or a communications port or other adapter.

[0162] Embodiments described herein may be entirely hardware, entirely software or including both hardware and software elements. In a preferred embodiment, the present invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

[0163] Embodiments may include a computer program product accessible from a computer-usable or computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. A computer-usable or computer readable medium may include any apparatus that stores, communicates, propagates, or transports the program for use by or in connection with the instruction

execution system, apparatus, or device. The medium can be magnetic, optical, electronic, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. The medium may include a computer-readable storage medium such as a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk, etc.

[0164] Each computer program may be tangibly stored in a machine-readable storage media or device (e.g., program memory or magnetic disk) readable by a general or special purpose programmable computer, for configuring and controlling operation of a computer when the storage media or device is read by the computer to perform the procedures described herein. The inventive system may also be considered to be embodied in a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner to perform the functions described herein.

[0165] A data processing system suitable for storing and/or executing program code may include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code to reduce the number of times code is retrieved from bulk storage during execution. Input/output or I/O devices (including but not limited to keyboards, displays, pointing devices, etc.) may be coupled to the system either directly or through intervening I/O controllers.

[0166] Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems,

cable modem and Ethernet cards are just a few of the currently available types of network adapters.

[0167] As employed herein, the term "hardware processor subsystem" or "hardware processor" can refer to a processor, memory, software or combinations thereof that cooperate to perform one or more specific tasks. In useful embodiments, the hardware processor subsystem can include one or more data processing elements (e.g., logic circuits, processing circuits, instruction execution devices, etc.). The one or more data processing elements can be included in a central processing unit, a graphics processing unit, and/or a separate processor- or computing element-based controller (e.g., logic gates, etc.). The hardware processor subsystem can include one or more on-board memories (e.g., caches, dedicated memory arrays, read only memory, etc.). In some embodiments, the hardware processor subsystem can include one or more memories that can be on or off board or that can be dedicated for use by the hardware processor subsystem (e.g., ROM, RAM, basic input/output system (BIOS), etc.).

[0168] In some embodiments, the hardware processor subsystem can include and execute one or more software elements. The one or more software elements can include an operating system and/or one or more applications and/or specific code to achieve a specified result.

[0169] In other embodiments, the hardware processor subsystem can include dedicated, specialized circuitry that performs one or more electronic processing functions to achieve a specified result. Such circuitry can include one or more application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs), and/or programmable logic arrays (PLAs).

[0170] These and other variations of a hardware processor subsystem are also contemplated in accordance with embodiments of the present invention.

[0171] Reference in the specification to "one embodiment" or "an embodiment" of the present invention, as well as other variations thereof, means that a particular feature, structure, characteristic, and so forth described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, the appearances of the phrase "in one embodiment" or "in an embodiment", as well any other variations, appearing in various places throughout the specification are not necessarily all referring to the same embodiment. However, it is to be appreciated that features of one or more embodiments can be combined given the teachings of the present invention provided herein.

[0172] It will be understood that when an element is referred to as being "connected" or "coupled" to another element, it can be directly connected or coupled to the other element or intervening elements can be present. In contrast, when an element is referred to as being "directly connected" or "directly coupled" to another element, there are no intervening elements present.

[0173] It is to be appreciated that the use of any of the following "/", "and/or", and "at least one of", for example, in the cases of "A/B", "A and/or B" and "at least one of A and B", is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of both options (A and B). As a further example, in the cases of "A, B, and/or C" and "at least one of A, B, and C", such phrasing is intended to encompass the selection of the first listed option (A) only, or the selection of the second listed option (B) only, or the selection of the third listed option (C) only, or the selection of the first and the second listed options (A and B) only, or the selection of the first and third listed options (A and C) only, or the selection of the second and third listed options (B and C) only, or the

selection of all three options (A and B and C). This may be extended for as many items listed.

[0174] The foregoing is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the invention disclosed herein is not to be determined from the Detailed Description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that the embodiments shown and described herein are only illustrative of the present invention and that those skilled in the art may implement various modifications without departing from the scope and spirit of the invention. Those skilled in the art could implement various other feature combinations without departing from the scope and spirit of the invention. Having thus described aspects of the invention, with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims.

WHAT IS CLAIMED IS:

1. A computer implemented method of retrieving similar multivariate time series segments, comprising:

extracting (920) a long feature vector and a short feature vector from a time series segment;

converting (930) the long feature vector into a long binary code;

converting (930) the short feature vector into a short binary code;

obtaining (940) a subset of long binary codes from a binary dictionary storing dictionary long codes based on the short binary codes;

calculating (950) similarity measure for each pair of the long feature vector with each dictionary long code;

identifying (960) a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes; and

retrieving (970) a predetermined number of time series segments associated with the predetermined number of dictionary long codes.

2. The computer implemented method as recited in claim 1, further comprising displaying the predetermined number of time series segments to a user.

3. The computer implemented method as recited in claim 1, wherein the long feature vector and the short feature vector are extracted from the time series segments using a long short term memory (LSTM).

4. The computer implemented method as recited in claim 3, wherein the long feature vector is converted into a long binary code by checking the signs of all entries in the feature vector.

5. The computer implemented method as recited in claim 4, wherein the short feature vector is converted into a short binary code by a linear mapping.

6. The computer implemented method as recited in claim 5, further comprising classifying the short binary codes to a class.

7. The computer implemented method as recited in claim 6, wherein classifying involves computing the probability of the short binary code belong to each of a plurality of labels associated with the time series segments.

8. A processing system for retrieving similar multivariate time series segments, comprising:

one or more processors (1004);

memory coupled to the one or more processors (1024);

a long feature extractor (310) stored in memory, wherein the long feature extractor is configured to extract (920) a long feature vector from a time series segment;

a short feature extractor (340) stored in memory, wherein the short feature extractor is configured to convert (920) a long feature generated by the long feature extractor (310) into a shorter length feature through a linear mapping;

a long binary extractor (320) stored in memory, wherein the long binary extractor is configured to convert (930) a long feature from the long feature extractor into a long binary code having the same length as the long feature;

a short binary extractor (350) stored in memory, wherein the short binary extractor is configured to convert (930) a short feature from the short feature extractor into a short binary code having the same length as the short feature; and

a similarity comparator (1130) stored in memory, wherein the similarity comparator is configured to calculate a pairwise similarity (950) between a long binary code extracted from the query and all long binary codes retrieved from a dictionary, and identify (960) a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes.

9. The processing system as recited in claim 8, wherein the short feature from the short feature extractor into a short binary code having the same length as the short feature by checking the sign of the entries in a short feature vector.

10. The processing system as recited in claim 8, wherein the similarity comparator is configured to retrieve a predetermined number of time series segments associated with the predetermined number of dictionary long codes, and display the predetermined number of time series segments to a user.

11. The processing system as recited as recited in claim 10, wherein the long feature vector and the short feature vector are extracted from the time series segments using a long short term memory (LSTM).

12. The processing system as recited as recited in claim 11, wherein the long feature vector is converted into a long binary code by checking the signs of all entries in the feature vector.

13. The processing system as recited as recited in claim 12, wherein the short feature vector is converted into a short binary code by a linear mapping.

14. The processing system as recited as recited in claim 13, wherein the short binary extractor is further configured to classifying the short binary codes to a class.

15. A computer program product for retrieving similar multivariate time series segments, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith, the program instructions executable by a computer to cause the computer to perform a method comprising:

extracting (920) a long feature vector and a short feature vector from a time series segment;

converting (930) the long feature vector into a long binary code;

converting (930) the short feature vector into a short binary code;

obtaining (940) a subset of long binary codes from a binary dictionary storing dictionary long codes based on the short binary codes;

calculating (950) similarity measure for each pair of the long feature vector with each dictionary long code;

identifying (960) a predetermined number of dictionary long codes having the similarity measures indicting a closest relationship between the long binary codes and dictionary long codes; and

retrieving (970) a predetermined number of time series segments associated with the predetermined number of dictionary long codes.

16. The computer program product as recited in claim 15, further comprising displaying the predetermined number of time series segments to a user.

17. The computer program product as recited in claim 15, wherein the long feature vector and the short feature vector are extracted from the time series segments using a long short term memory (LSTM).

18. The computer program product as recited in claim 17, wherein the long feature vector is converted into a long binary code by checking the signs of all entries in the feature vector.

19. The computer program product as recited in claim 18, wherein the short feature vector is converted into a short binary code by a linear mapping.

20. The computer program product as recited in claim 19, further comprising classifying the short binary codes to a class, wherein classifying involves computing the probability of the short binary code belong to each of a plurality of labels associated with the time series segments.

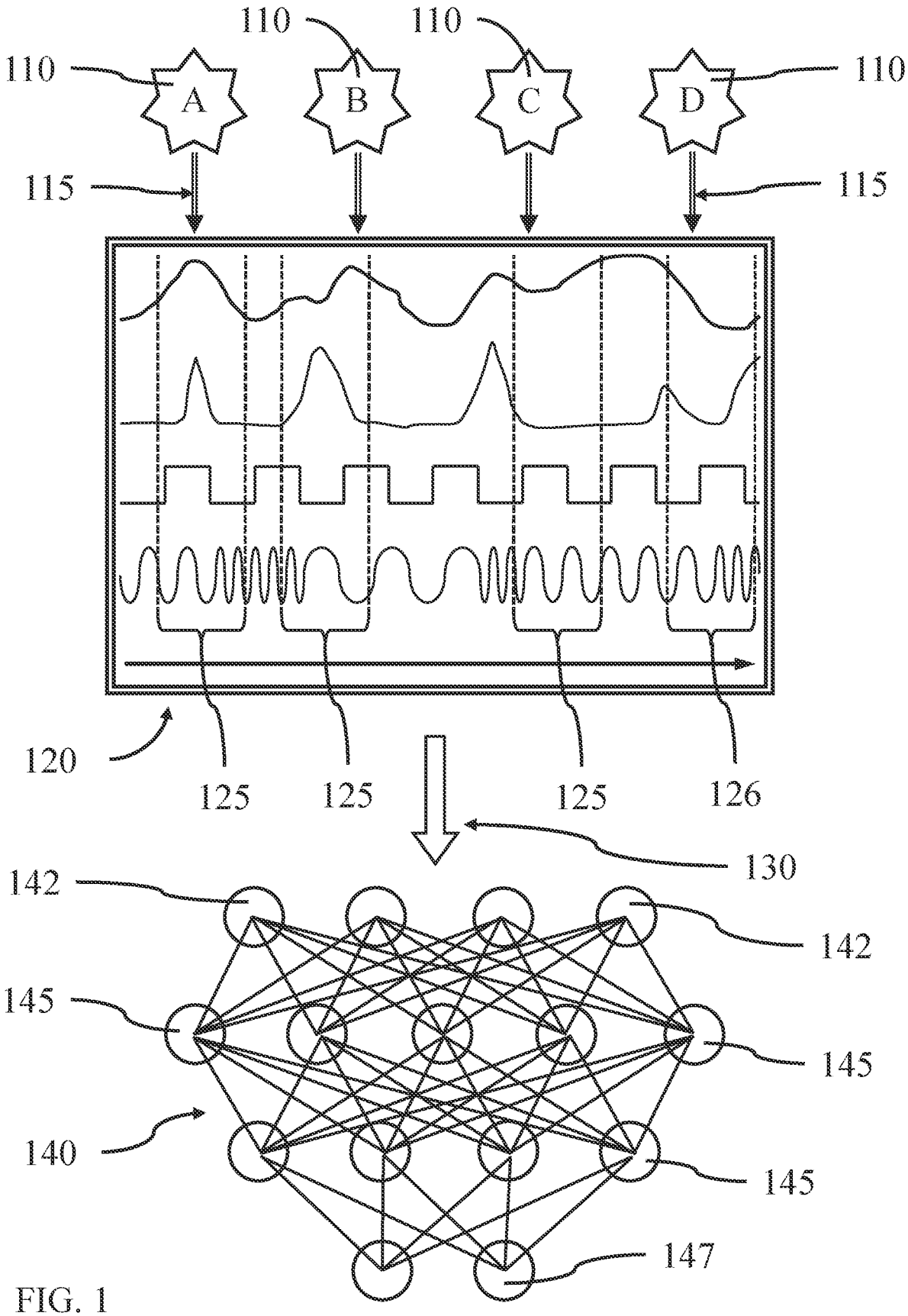
110
110
110
110
A
B
C
D
115
115
120
125
125
125
126
130
142
142
145
145
140
145
147

FIG. 1

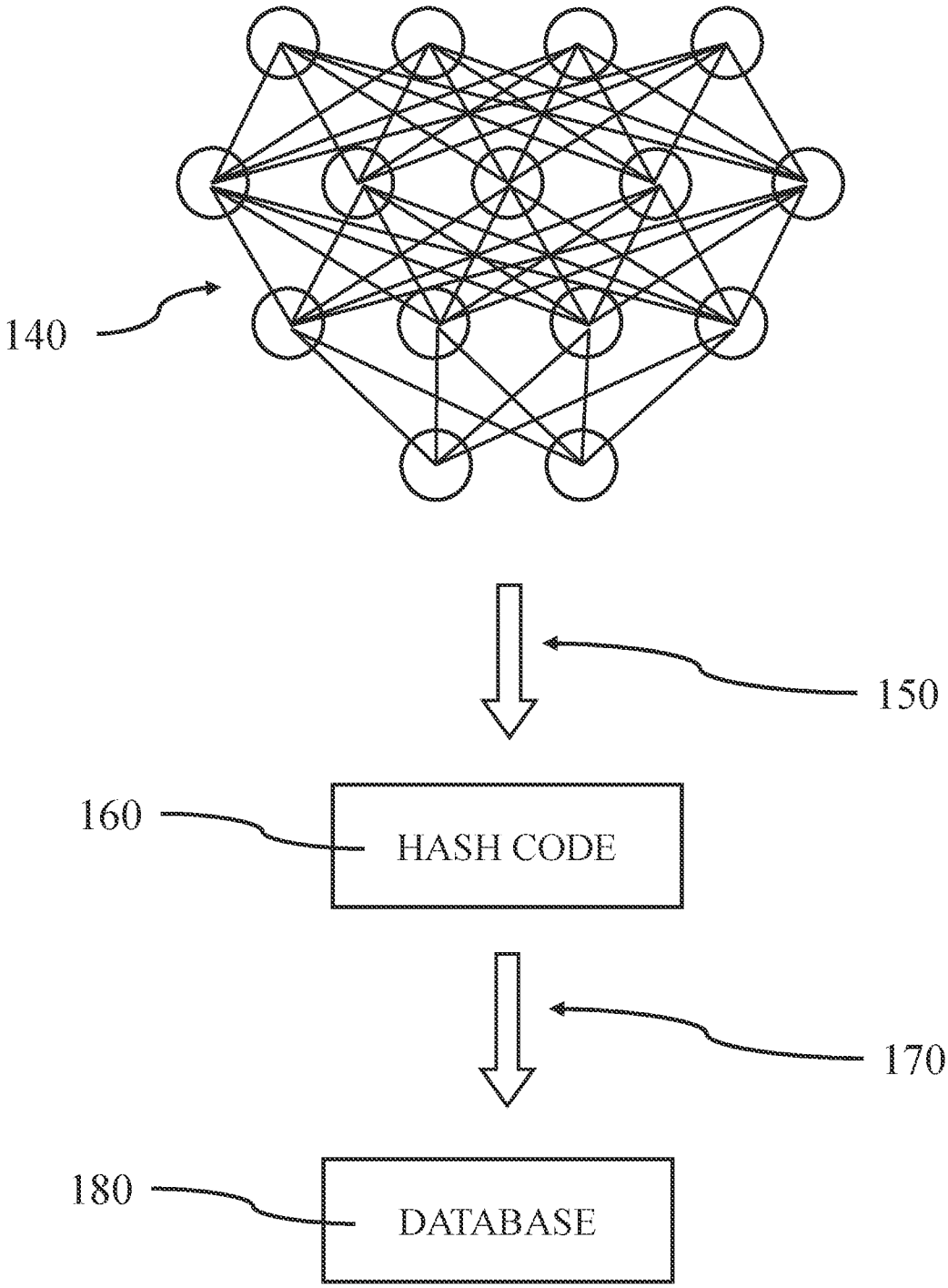

FIG. 2

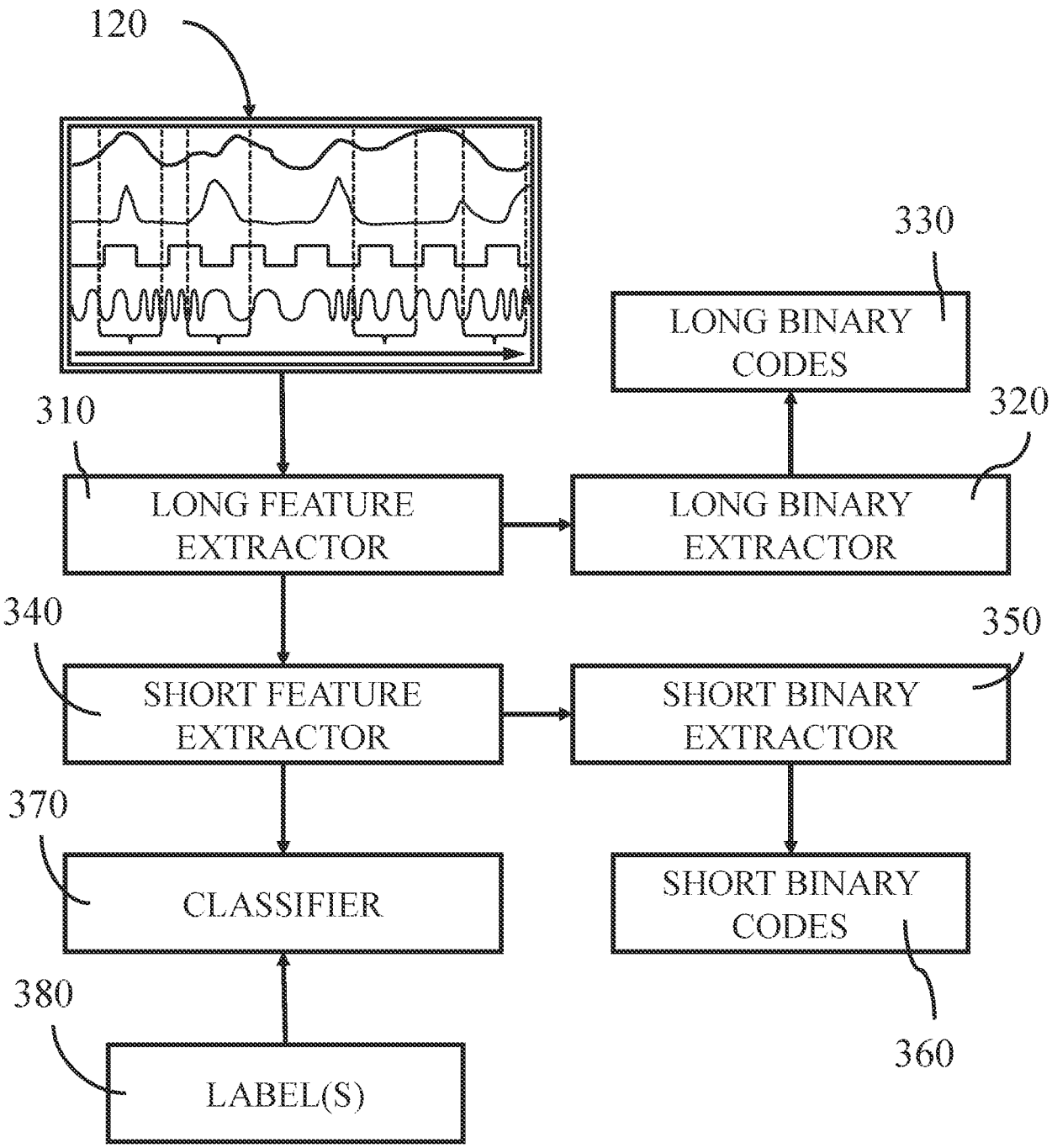
120
330
LONG BINARY CODES
310
320
LONG FEATURE EXTRACTOR
LONG BINARY EXTRACTOR
340
350
SHORT FEATURE EXTRACTOR
SHORT BINARY EXTRACTOR
370
CLASSIFIER
SHORT BINARY CODES
380
360
LABEL(S)

FIG. 3

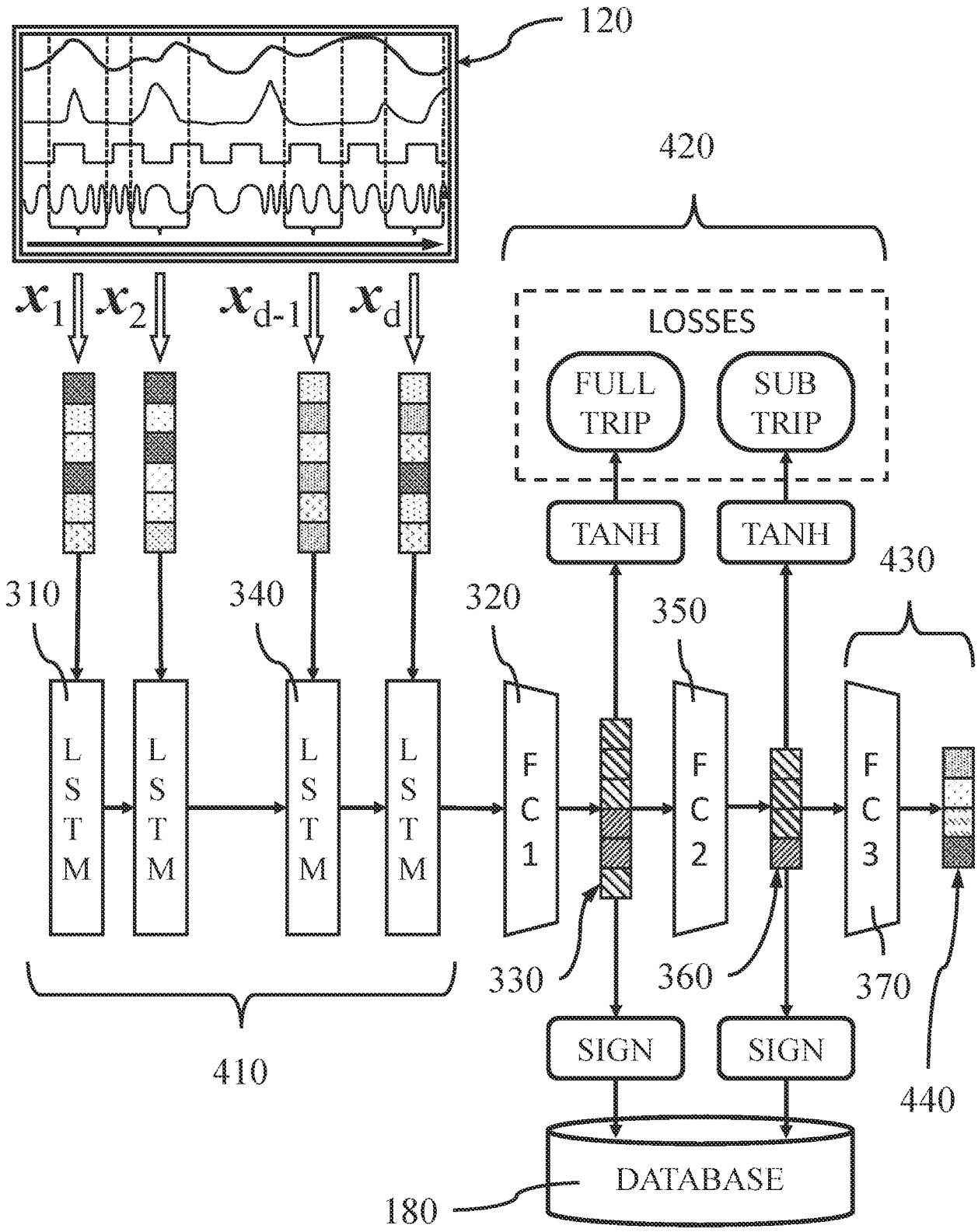
120
420
x_1
x_2
x_{d-1}
x_d
LOSSES
FULL TRIP
SUB TRIP
TANH
TANH
310
340
320
350
430
LSTM
LSTM
LSTM
LSTM
FC1
FC2
FC3
330
360
370
410
SIGN
SIGN
440
DATABASE
180

FIG. 4

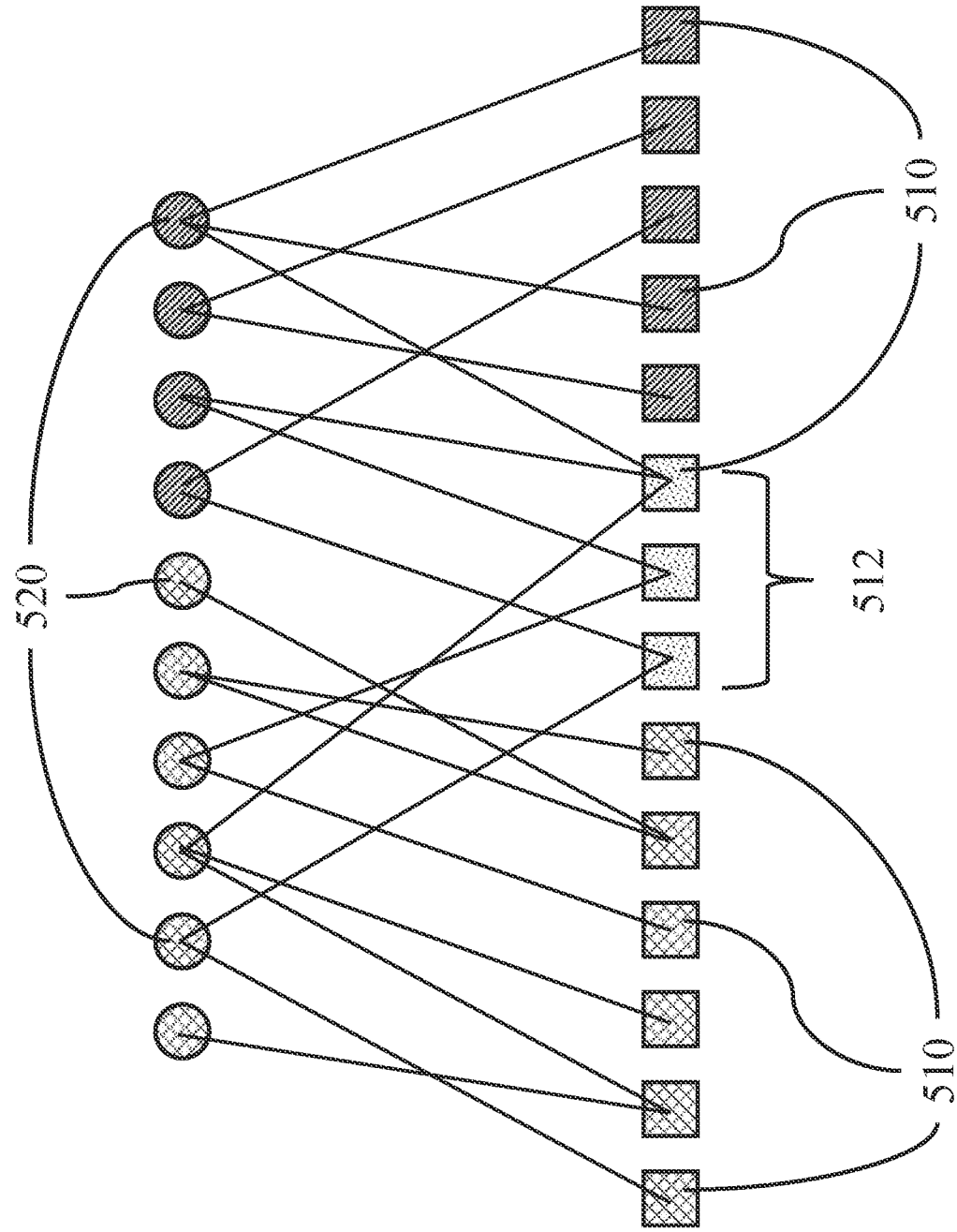

FIG. 5

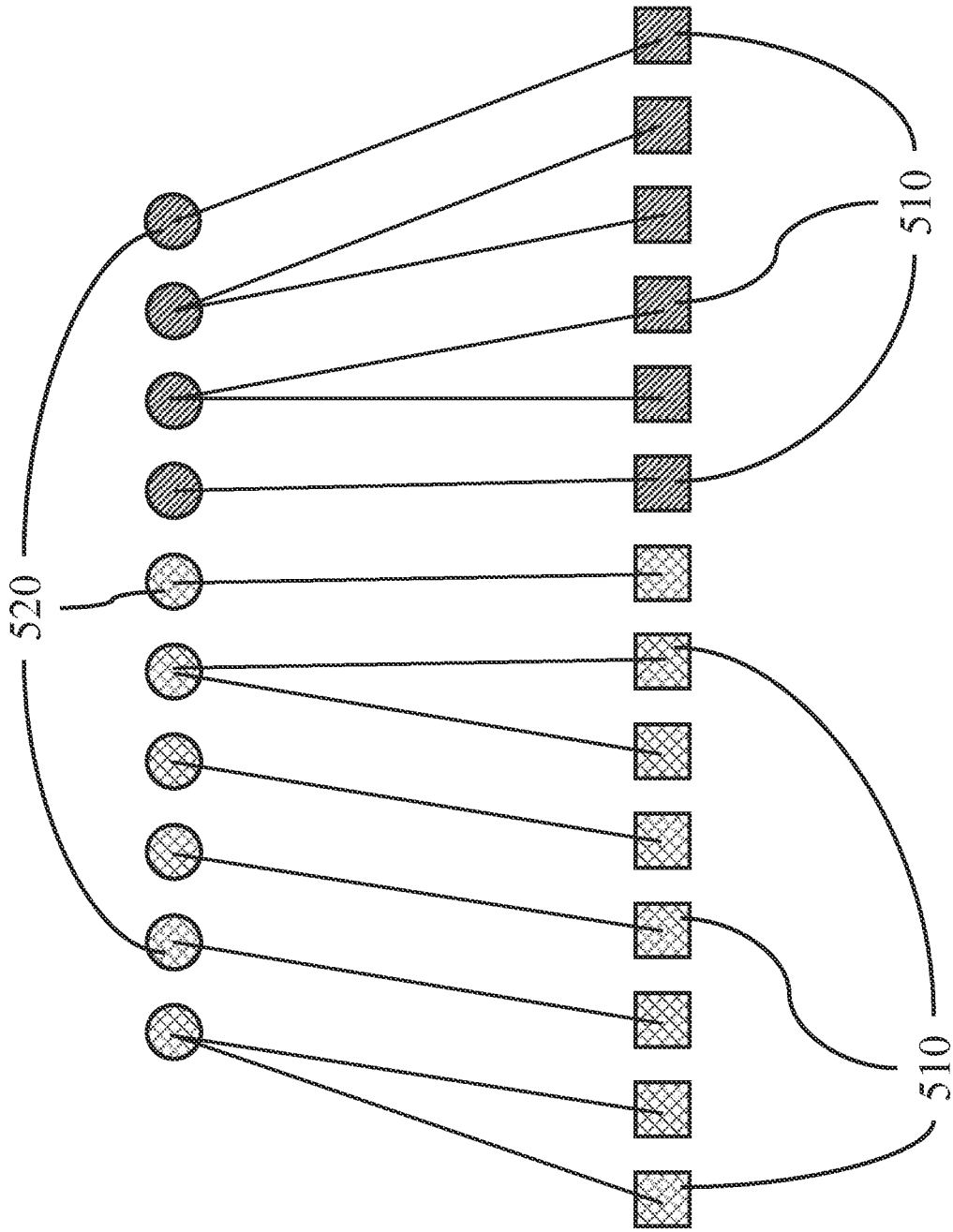

FIG. 6

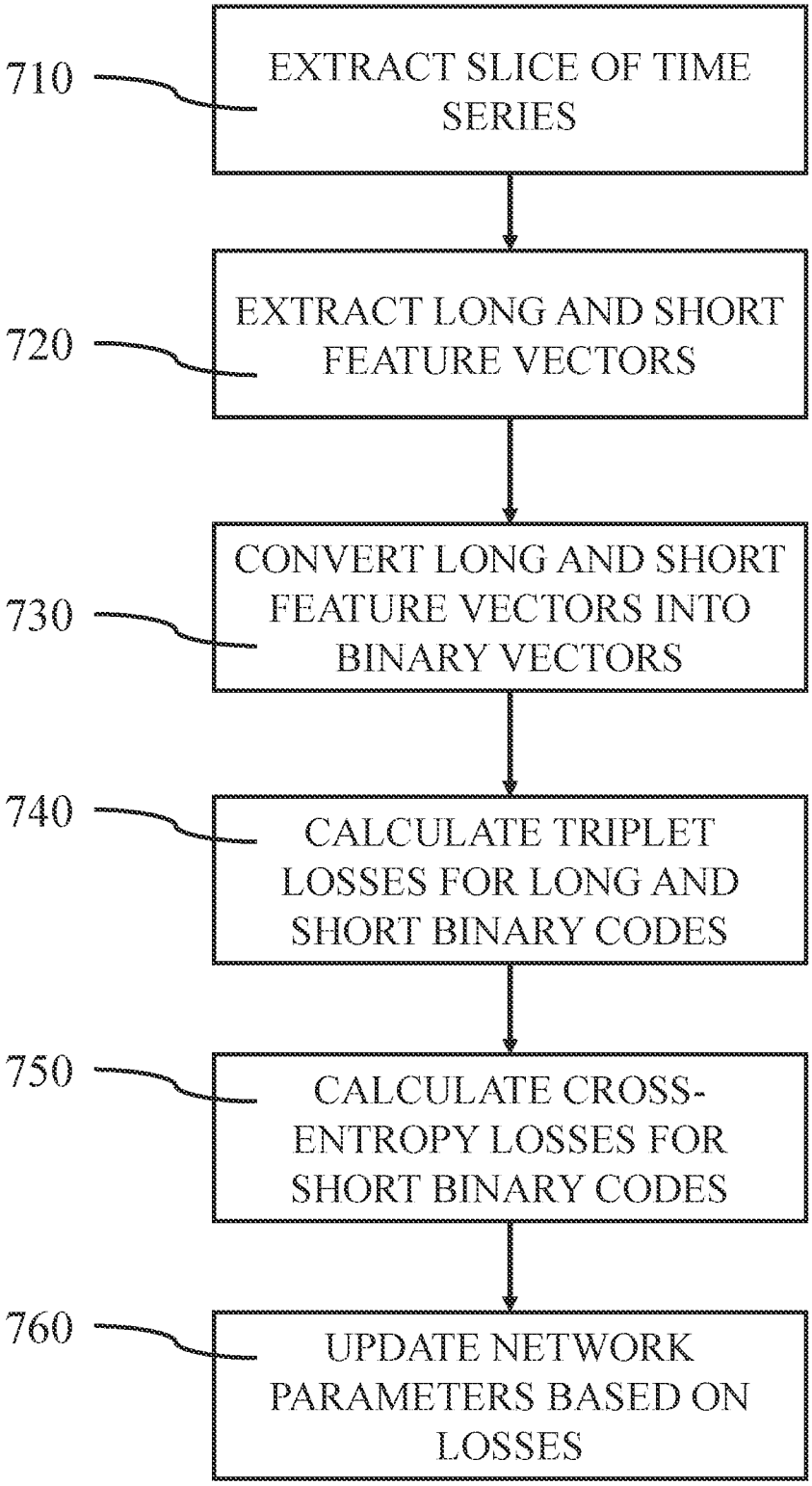
710
EXTRACT SLICE OF TIME SERIES
720
EXTRACT LONG AND SHORT FEATURE VECTORS
730
CONVERT LONG AND SHORT FEATURE VECTORS INTO BINARY VECTORS
740
CALCULATE TRIPLET LOSSES FOR LONG AND SHORT BINARY CODES
750
CALCULATE CROSS-ENTROPY LOSSES FOR SHORT BINARY CODES
760
UPDATE NETWORK PARAMETERS BASED ON LOSSES

FIG. 7

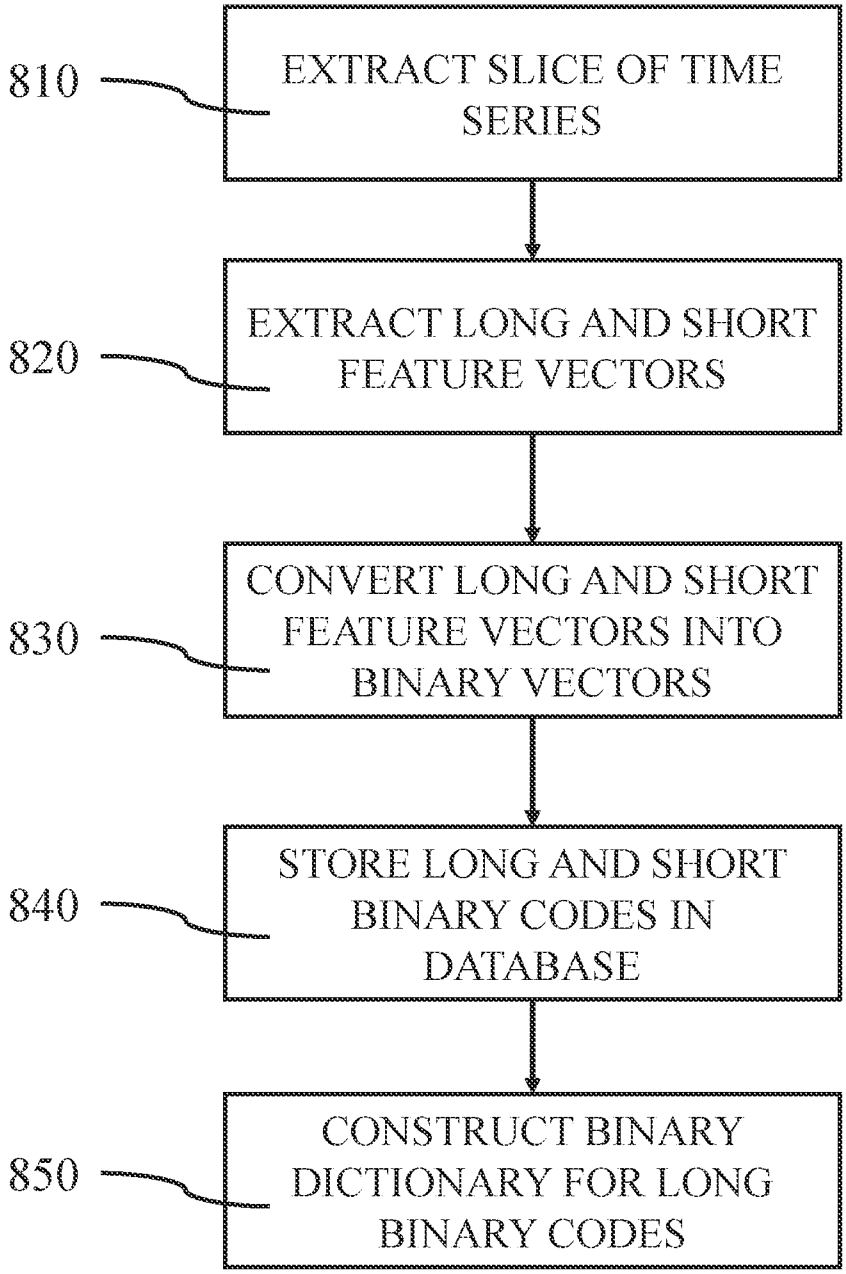
810
EXTRACT SLICE OF TIME SERIES
820
EXTRACT LONG AND SHORT FEATURE VECTORS
830
CONVERT LONG AND SHORT FEATURE VECTORS INTO BINARY VECTORS
840
STORE LONG AND SHORT BINARY CODES IN DATABASE
850
CONSTRUCT BINARY DICTIONARY FOR LONG BINARY CODES

FIG. 8

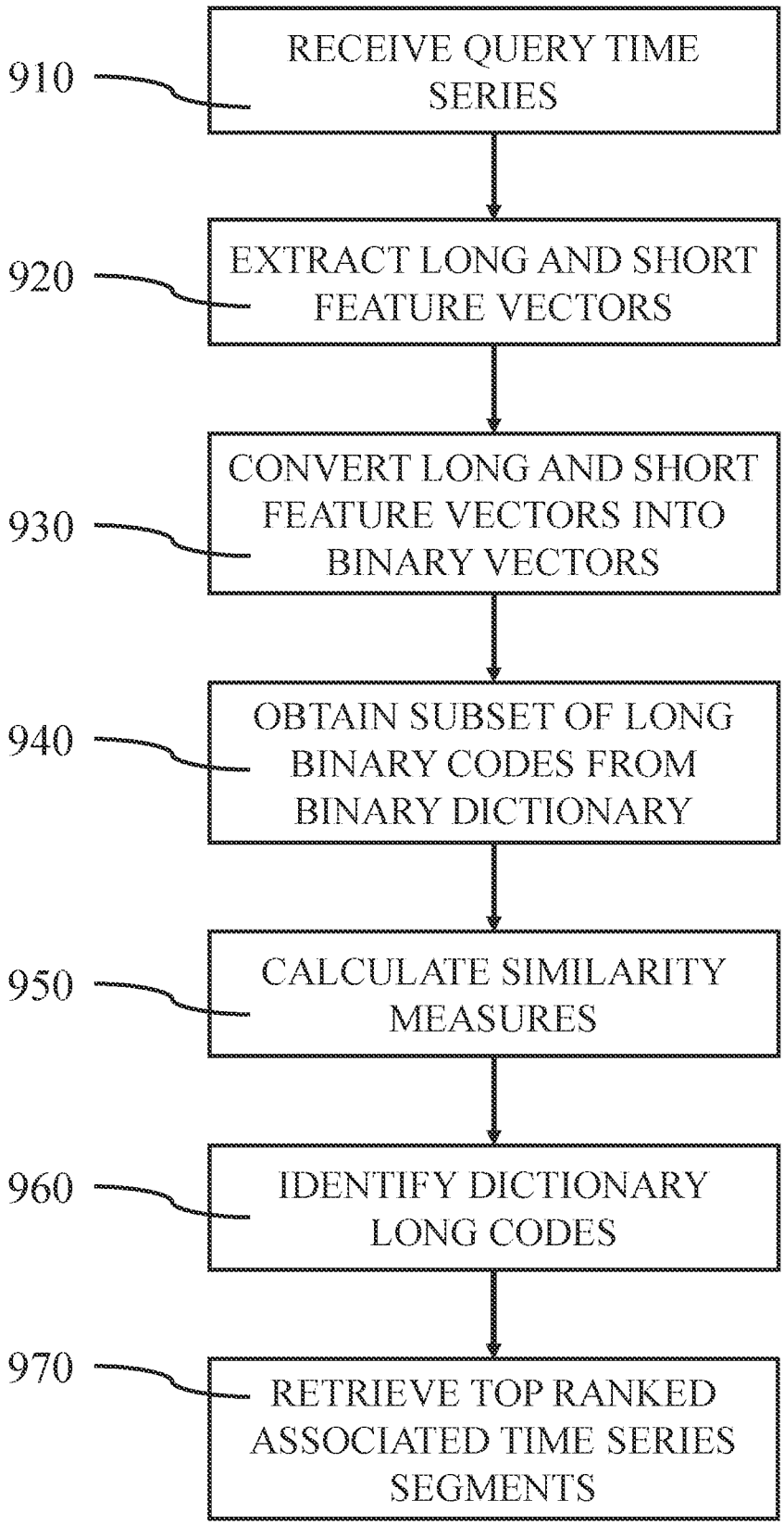
910
RECEIVE QUERY TIME SERIES
920
EXTRACT LONG AND SHORT FEATURE VECTORS
930
CONVERT LONG AND SHORT FEATURE VECTORS INTO BINARY VECTORS
940
OBTAIN SUBSET OF LONG BINARY CODES FROM BINARY DICTIONARY
950
CALCULATE SIMILARITY MEASURES
960
IDENTIFY DICTIONARY LONG CODES
970
RETRIEVE TOP RANKED ASSOCIATED TIME SERIES SEGMENTS

FIG. 9

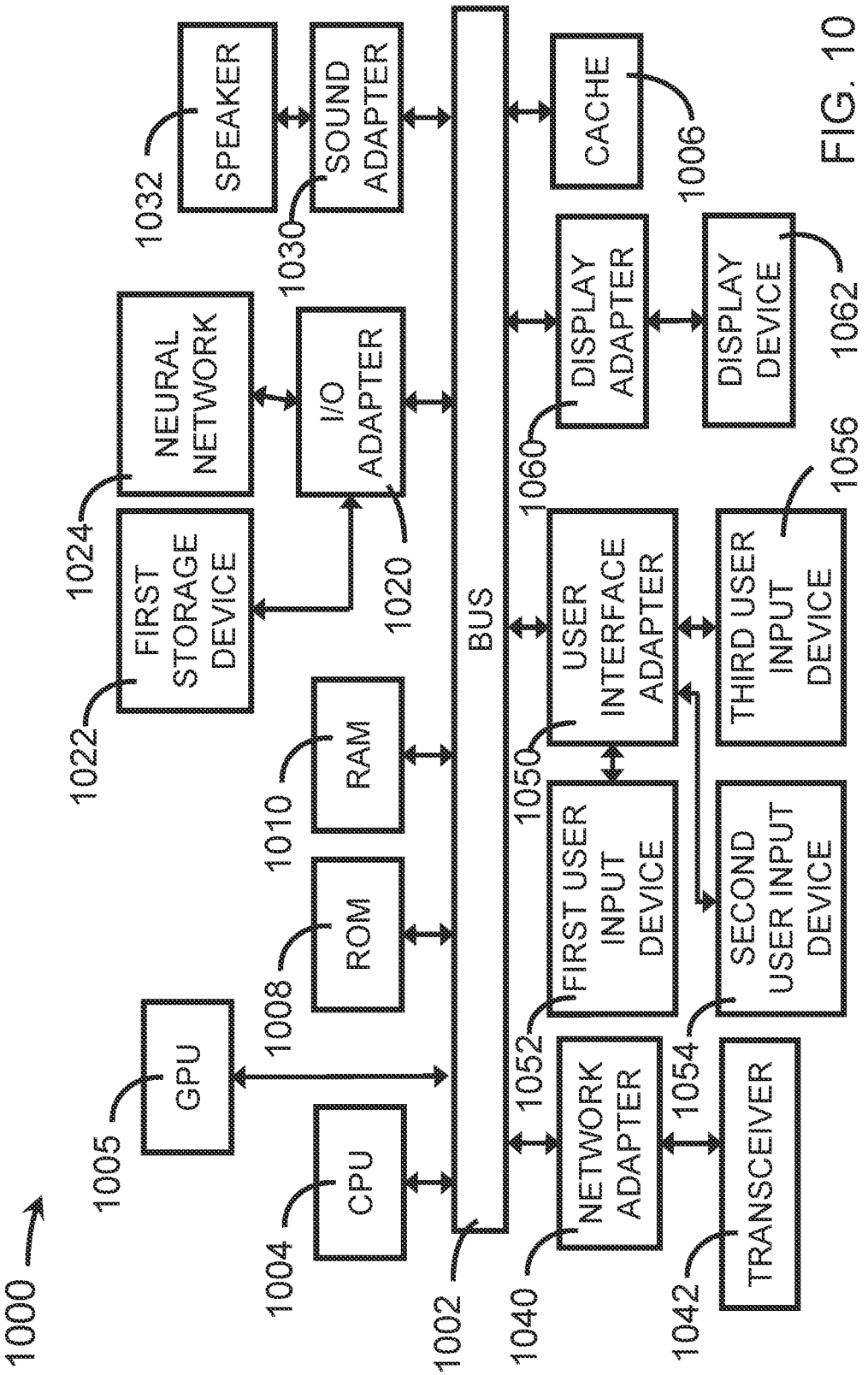
1000
1005
GPU
1004
CPU
1008
ROM
1010
RAM
1022
FIRST STORAGE DEVICE
1024
NEURAL NETWORK
1020
I/O ADAPTER
1032
SPEAKER
1030
SOUND ADAPTER
1002
BUS
1006
CACHE
1060
DISPLAY ADAPTER
1062
DISPLAY DEVICE
1050
USER INTERFACE ADAPTER
1056
THIRD USER INPUT DEVICE
1052
FIRST USER INPUT DEVICE
1054
SECOND USER INPUT DEVICE
1040
NETWORK ADAPTER
1042
TRANSCEIVER

FIG. 10

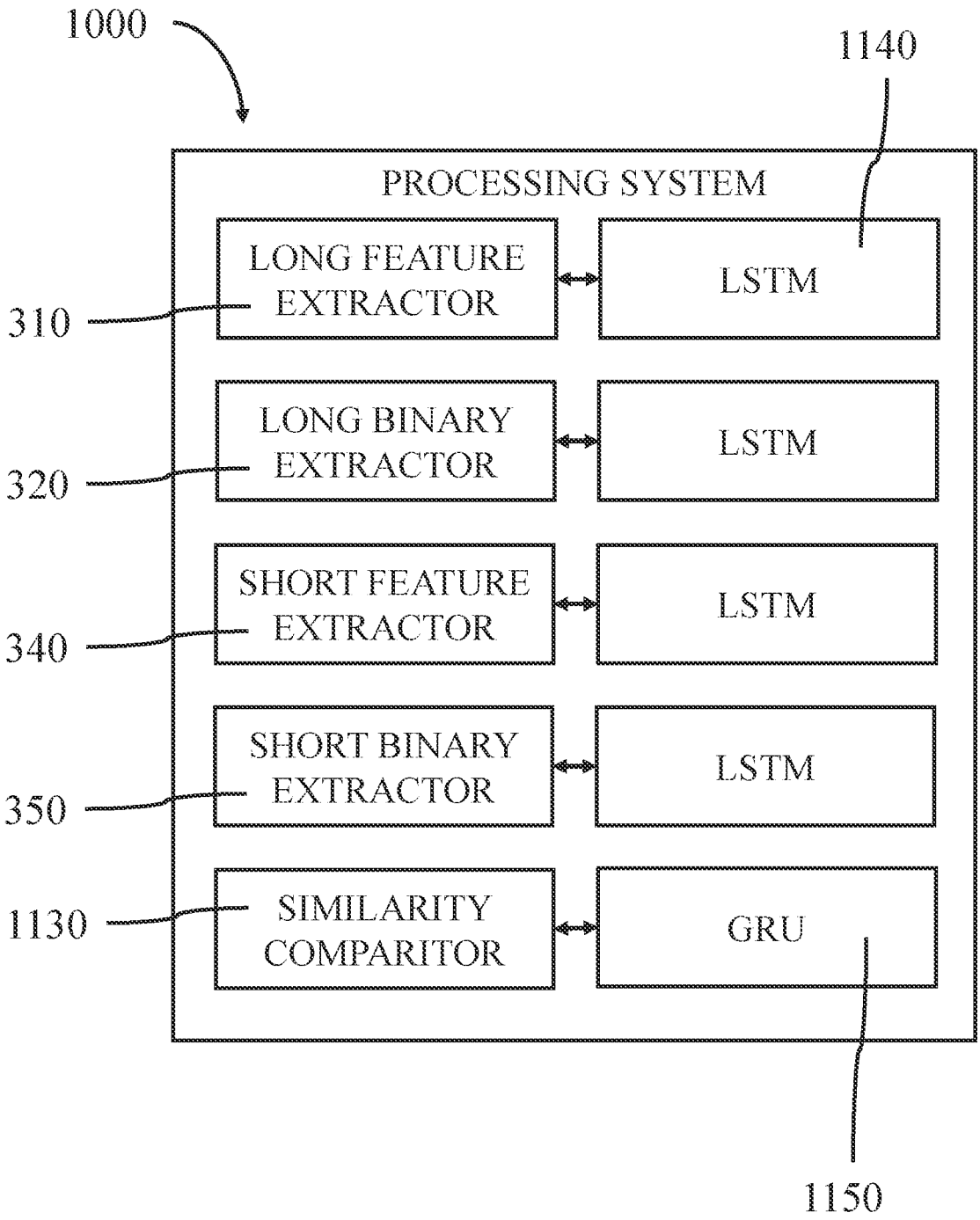
1000
1140
PROCESSING SYSTEM
LONG FEATURE EXTRACTOR
LSTM
310
LONG BINARY EXTRACTOR
LSTM
320
SHORT FEATURE EXTRACTOR
LSTM
340
SHORT BINARY EXTRACTOR
LSTM
350
SIMILARITY COMPARITOR
GRU
1130
1150

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2021/040081

A. CLASSIFICATION OF SUBJECT MATTER

G06F 16/28(2019.01)i; **G06F 16/25**(2019.01)i; **G06F 16/22**(2019.01)i; **G06F 16/2458**(2019.01)i; **G06N 3/08**(2006.01)i; **G06N 3/04**(2006.01)i; **G06N 20/00**(2019.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 16/28(2019.01); G06F 15/18(2006.01); G06F 17/30(2006.01); G06K 9/00(2006.01); G06K 9/62(2006.01); G06N 3/04(2006.01); G06N 3/08(2006.01); G10L 15/16(2006.01); G10L 15/22(2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: time series segment, feature vector, binary code, short, long, similarity, dictionary

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DAMIAN J. MATUSZEWSKI et al., `A short feature vector for image matching: The Log-Polar Magnitude feature descriptor`, PLOS ONE, 30 November 2017, pp. 1-21 pages 1-8; and figure 3	1-20
A	US 2020-0152179 A1 (SRI INTERNATIONAL) 14 May 2020 (2020-05-14) paragraph [0025]; claims 1, 6-7; and figures 5-6	1-20
A	US 2011-0289026 A1 (ANITHA KANNAN et al.) 24 November 2011 (2011-11-24) paragraphs [0044]-[0048], [0059]; claims 1-3; and figure 2	1-20
A	CN 111091080 A (GUIZHOU POWER GRID CO., LTD.) 01 May 2020 (2020-05-01) claims 1, 4-6; and figure 2	1-20
A	WO 2019-176986 A1 (NEC CORPORATION) 19 September 2019 (2019-09-19) paragraphs [0018]-[0022], [0038]-[0040]; claims 1, 5-6; and figures 5-6	1-20

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"D" document cited by the applicant in the international application
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 October 2021	**20 October 2021**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**YANG, JEONG ROK**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-5709**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2021/040081

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 2020-0152179 A1	14 May 2020	US 10777188 B2	15 September 2020
US 2011-0289026 A1	24 November 2011	US 2013-0073592 A1	21 March 2013
		US 8417651 B2	09 April 2013
		US 8805753 B2	12 August 2014
CN 111091080 A	01 May 2020	None	
WO 2019-176986 A1	19 September 2019	US 2021-0050021 A1	18 February 2021