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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 February 2024 (29.02.2024)

WIPO | PCT

(10) International Publication Number
WO 2024/044153 A1

(51) International Patent Classification:
H04B 10/079 (2013.01) *H04J 14/02* (2006.01)
G06N 20/00 (2019.01)

(21) International Application Number:
PCT/US2023/030773

(22) International Filing Date:
22 August 2023 (22.08.2023)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

63/399,747	22 August 2022 (22.08.2022)	US
63/408,550	21 September 2022 (21.09.2022)	US
18/452,664	21 August 2023 (21.08.2023)	US

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

(72) Inventors: **CHENG, Wei**; 5 Arnold Drive, Princeton Junction, New Jersey 08550 (US). **NI, Jingchao**; 300 Bunn Drive, Princeton, New Jersey 08540 (US). **TONG, Liang**; 3434 Town Court South, Lawrenceville, New Jersey 08648 (US). **CHEN, Haifeng**; 11 Blackhawk Court, West Windsor, New Jersey 08550 (US). **ZHANG, Yizhou**; PO Box 18166, Los Angeles, California 90018 (US).

(74) Agent: **BITETTO, James J.**; 401 Broadhollow Road, Suite 402, Melville, New York 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, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, 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, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, 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, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE,

(54) Title: SNR DETECTION WITH FEW-SHOT TRAINED MODELS

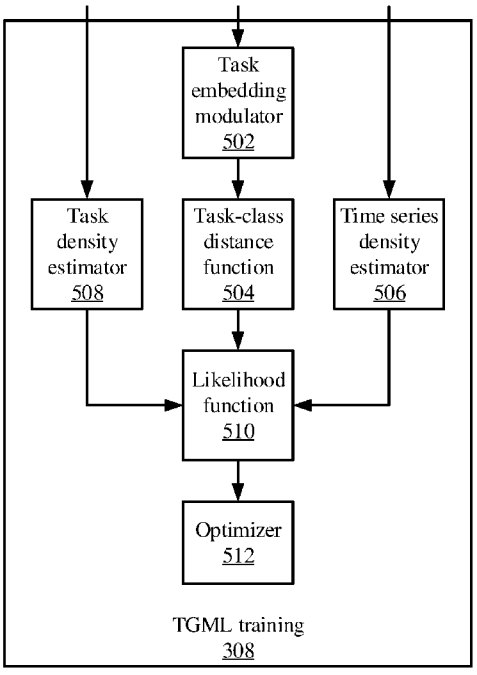

FIG. 5

(57) Abstract: Methods and systems for training a model include determining (604) class prototypes of time series samples from a training dataset. A task corresponding to the time series samples is encoded (606) using the class prototypes and a task-level configuration. A likelihood value is determined (608) based on outputs of a time series density model, a task-class distance from a task embedding model, and a task density model. Parameters of the time series density model, the task embedding model, and the task density model are adjusted (610) responsive to the likelihood value.

WO 2024/044153 A1

[Continued on next page]

SI, SK, SM, 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))*

SNR DETECTION WITH FEW-SHOT TRAINED MODELS

RELATED APPLICATION INFORMATION

[0001] This application claims priority to U.S. Patent Application No. 63/399,747, filed on August 22, 2022, U.S. Patent Application No. 63/408,550, filed on September 21, 2022, and U.S. Patent Application No. 18/452,664 filed on August 21, 2023, each incorporated herein by reference in its entirety.

BACKGROUND

Technical Field

[0002] The present invention relates to optical networking and, more particularly, to detection of signal-to-noise ratio (SNR) using machine learning.

Description of the Related Art

[0003] Optical networking encodes signals in light to transmit information in telecommunications networks. Such networks may include local area networks (LANs) or wide area networks (WANs), and may be distributed across metropolitan and regional areas as well as being used for long-distance national, international, and transoceanic networks. Optical networking makes use of optical amplifiers, lasers or light emitting diodes (LEDs), and wave division multiplexing (WDM) equipment to transmit large amounts of data across fiber-optic cables. Optical networking can provide high bandwidth and is upgradeable, as new endpoint equipment can generally make use of existing fiber-optic cables.

[0004] To ensure transmission quality in optical networks, signals may be monitored to identify the system status of optical transceivers. However, user equipment may differ from the environment that is used as a baseline, so that models used for SNR

determination may not be applicable to a particular testing environment. Manual measurement of SNR in a given environment, meanwhile, is time-consuming and uses specialized equipment.

SUMMARY

[0005] A method for training a model includes determining class prototypes of time series samples from a training dataset. A task corresponding to the time series samples is encoded using the class prototypes and a task-level configuration. A likelihood value is determined based on outputs of a time series density model, a task-class distance from a task embedding model, and a task density model. Parameters of the time series density model, the task embedding model, and the task density model are adjusted responsive to the likelihood value.

[0006] A system for training a model includes a hardware processor and a memory that stores a computer program. When executed by the hardware processor, the computer program causes the hardware processor to determine class prototypes of time series samples from a training dataset, to encode a task corresponding to the time series samples using the class prototypes and a task-level configuration, to determine a likelihood value based on outputs of a time series density model, a task-class distance from a task embedding model, and a task density model, and to adjust parameters of the time series density model, the task embedding model, and the task density model responsive to the likelihood value.

[0007] 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

[0008] The disclosure will provide details in the following description of preferred embodiments with reference to the following figures wherein:

[0009] FIG. 1 is a diagram of an exemplary optical network, where properties of optical signals transmitted over the optical network depend on the configuration of the optical network, in accordance with an embodiment of the present invention;

[0010] FIG. 2 is a block diagram of a system that trains a model for use in a target environment, distinct from an experimental environment, in accordance with an embodiment of the present invention;

[0011] FIG. 3 is a block/flow diagram of a method of training a model, in accordance with an embodiment of the present invention;

[0012] FIG. 4 is a block diagram of an exemplary inference network, in accordance with an embodiment of the present invention;

[0013] FIG. 5 is a block diagram of a system that trains a model, in accordance with an embodiment of the present invention;

[0014] FIG. 6 is a block/flow diagram of a method of training a model, in accordance with an embodiment of the present invention;

[0015] FIG. 7 is a block/flow diagram of a method of classifying time series data, in accordance with an embodiment of the present invention;

[0016] FIG. 8 is a block diagram of a computing system that can train and use a model and that can perform network management, in accordance with an embodiment of the present invention;

[0017] FIG. 9 is an exemplary neural network architecture that can be used in a policy model, in accordance with an embodiment of the present invention; and

[0018] FIG. 10 is an exemplary deep neural network architecture that can be used in a policy model, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0019] Signal-to-noise ratio (SNR) in a given test environment may be measured using a trained machine learning model. However, because a training environment used to generate training data for the machine learning model may differ substantially from the test environment, such a limited model may give inaccurate results. The environments may have substantially different configurations, varying in terms of the network topology (e.g., ring topologies, star topologies, grid topologies, etc.), the number of nodes in the network, the quality and type of optical fiber, and so on.

[0020] Due to the large variety of different configurations, training data for any given configuration may be limited. Further, different configurations may represent different domains, with each domain having different domain-specific considerations. Testing data from previously unseen domains may be particularly challenging, and there may be configuration parameters that differ in ways that were not anticipated during training.

[0021] A meta-learning framework may be used to quickly adapt a trained model to a new data distribution. During training, a certain number of training sets may be generated under different respective conditions. A machine learning model may be trained by iterating over the training sets. At testing time, a relatively small number of labeled examples may be generated at the target environment and may be used to fine-tune the model. The fine-tuned model may then be used for future predictions in the target environment. The generation of a relatively small number of labeled examples in the target environment is more feasible than generating an entire training corpus that would be suitable for training the entire model.

[0022] Referring now to FIG. 1, an exemplary optical networking environment 100 is shown. The environment includes an optical line terminal (OLT) 102 and one or more optical network terminals (ONTs) 106. Fiber-optic cables 108 connect the devices to one another. A single cable 108 leaving the OLT 102 may carry multiple different optical signals, for example being combined on a single physical medium with wavelength division multiplexing (WDM). In some cases, multiple cables 108 may leave the OLT 102.

[0023] A cable 108 from the OLT 102 leads to an optical splitter 104. In some cases, the optical splitter may use wavelength-selective splitting to divide the combined optical signal from the OLT 102 into different signals having different respective wavelengths. In some cases the optical splitter may be insensitive to wavelength, in which case filtering may be performed at the ONTs 106 to remove unwanted wavelengths. The ONTs 106 receive optical signals from the splitter 104 by way of further fiber optic cables 108. The ONTs 106 convert the optical signals to electrical signals suitable for processing by downstream network hardware (not shown).

[0024] The signals may include a variety of different types of modulation and multiplexing. For example, in addition to WDM, polarization multiplexing may be used to transmit multiple signals on each wavelength with orthogonal optical polarizations. Information may be modulated onto the optical signals using any appropriate modulation scheme or schemes, such as phase modulation, frequency modulation, polarization modulation, amplitude modulation, spatial modulation, and diffraction modulation.

[0025] During the trip from the OLT 102 to the ONT 106, noise may accumulate in the transmitted signals from a variety of sources. These sources may include interference from, e.g., outside signals, inter-symbol interference, chromatic dispersion,

polarization drift, imperfections in the transmission medium, and index of refraction mismatches. The ONTs 106 may include equalization hardware that compensates for errors that may accumulate during transmission over the fiber-optic cables 108. However, measurement of SNR remains a common measure of optical network performance and imposes limits on the amount of information that can be transmitted in the environment 100.

[0026] SNR information can be measured within the environment, for example by comparing measurements of signals at the OLT 102 and at the ONT 106. Such measurements may be performed manually, but are relatively time consuming. In contrast, measurements of received signal information at the ONT 106 may be used as input to a trained machine learning model. The model may be trained on a dataset that includes received signals labeled according to actual measured SNR values. The model may thereby be used to generate SNR values for the input received signal information without the need for manual measurement.

[0027] However, because the configuration of environment 100 may differ substantially from the configuration(s) of an experimental environment that was used to generate the training data, such a machine learning model may not produce correct outputs. Meta-learning may therefore be used to generate a machine learning model that can be fine-tuned for a target environment with a relatively small number of labeled examples.

[0028] The signals received by the ONTs 106 may be monitored and tracked to identify the system status. The signals may include data obtained after polarization decomposition and data obtained after digital signal processing (DSP). For example, a snapshot of such a signal may record twenty thousand complex numbers (e.g., ten thousand signals for each of the x and y polarizations). The snapshots may be collected

at subsequent time steps to record a time series. In real analysis, this data is transformed to a multivariate time series by dividing the complex plane into 10×10 grids. The number of signals seen in each grid is counted, which continuously changes over time and hence contributes a dimension of the multivariate time series data. The total number of dimensions is the number of grids.

[0029] A machine learning model may be trained 204/206 on the multivariate time series data from an optical network. The trained model may be tuned to a particular environment, distinct from the experimental environment from which most of the training information was derived, to automatically infer the SNR of newly recorded time series information.

[0030] Referring now to FIG. 2, a meta-learning approach to SNR measurement is shown. Training data is collected from *n* environments 202, each of which may have a different respective configuration for an optical network. The training data may include a set of multivariate time series measurements that track multiple data points over time. The time series may be associated with respective labels, indicating an SNR value. Any appropriate SNR measurement may be used. In one particular example, the SNR value may range from 0, indicating unacceptable SNR, to 35, indicating a very good SNR.

[0031] Training data 202 is used as input to a meta-training process 204, which trains an appropriate machine learning model. The training process may include a backpropagation process, whereby training examples are compared to predictions made by the model, and whereby discrepancies are used to adjust the parameters of the model. The meta-training 204 generates a trained baseline model.

[0032] Fine-tuning 206 is then performed using target environment data 208. The target environment data 208 may be measured by any appropriate means, such as by manual measurement of the target environment. A smaller number of labeled examples

may be available in the target environment data 208 than in the training data 202. Fine-tuning 206 adapts the trained baseline model to a trained target model that is equipped to handle the target environment by generating new prototypes in a meta-testing support set, to help finish classification on the meta-testing's query set. In some embodiments, the fine-tuning 206 may not itself update parameters of the trained model. The fine-tuned model is then used in testing 210, where a new task is performed based on new inputs.

[0033] Each environment may be treated as a different task operating on a different distribution of features. Modeling a mixture of task distributions and detecting novel tasks are two aspects of the same problem, density estimation on task instances. The present model may therefore be implemented as a hierarchical Gaussian mixture based task generative model, which may be used to explicitly model the generative process of task instances. Task generation may be extended to a new process specified by a hierarchy of Gaussian mixture distributions. The model generates a task embedding from a task-level Gaussian mixture and uses it to define the task-conditioned mixture probabilities for a class-level Gaussian mixture, from which data samples are drawn. A Gibbs distribution may be used to underlie the class-level Gaussian mixture, to allow realistic classes per task.

[0034] The present model may be implemented as an encoder-agnostic framework that is flexible to different domains. A metric-based meta-learning method introduces a small overhead to an encoder for parameterizing its distributions and is therefore efficient for large backbone networks. The model parameters are learned end-to-end by maximum likelihood estimation.

[0035] Meta-learning may use an episodic learning strategy, where the meta-training set $\mathcal{D}^{tr}$ includes a batch of episodes. Each episode may sample a task τ from a

distribution $p(\tau)$. Each task τ may have a support set $\mathcal{D}_\tau^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$ for training and a query set $\mathcal{D}_\tau^q = \left\{\left(\mathbf{x}_i^q, y_i^q\right)\right\}_{i=1}^{n_s}$ for testing, where the number of samples in the support set n_s is a relatively small number (e.g., between 10 and 20, though any number smaller than a number of training samples is contemplated) and where n_q is the number of samples in the query set. The data values in each set are made up of a time series $\mathbf{x}_i$ and an associated label y_i. In an N-way, K-shot Q-query task, the sets $\mathcal{D}_\tau^s$ and $\mathcal{D}_\tau^q$ include *N* classes, with *K* and *Q* samples per class respectively, such that $n_s = NK$ and $n_q = NQ$.

[0036] A base mode may be expressed as $f_\theta(\mathbf{x}_i^*) \rightarrow y_i^*$, where $*$ denotes *s* or *q*, and an adapted model may be expressed as $f_\theta(\cdot; \mathcal{D}_\tau^s)$. The training objective on τ seeks to minimize the average test error of the adapted model:

$$\mathbb{E}_{(\mathbf{x}_i^q, y_i^q) \in \mathcal{D}_\tau^q} \ell\left(y_i^q, f_\theta\left(\mathbf{x}_i^q; \mathcal{D}_\tau^s\right)\right)$$

where $\ell(\cdot;\cdot)$ is a loss function, such as the cross-entropy loss. The meta-training process aims to find the parameter θ that minimizes this error over all episodes in $\mathcal{D}^{tr}$. Then f_θ may be evaluated on every episode of a meta-test set $\mathcal{D}^{te}$ that samples a task from the same distribution $p(\tau)$. This distribution may be generalized to a mixture distribution that has multiple components $p_1(\tau), \ldots, p_r(\tau)$, and a test episode may sample a task either in or out of any component of $p(\tau)$, where *r* is a hyper-parameter denoting the number of components of the mixture distribution. Given the training tasks in $\mathcal{D}^{tr}$, the underlying density of $p(\tau)$ may be estimated so that, once a test task is given, it can be determined whether the test task is a novel task and f_θ can be adapted with accuracy.

[0037] The base model f_θ may be written as a combination of an encoder g_{θ_e} and a predictor h_{θ_p}, such as $f_\theta(\mathbf{x}_i^*) = h_{\theta_p}\left(g_{\theta_e}(\mathbf{x}_i^*)\right)$. A metric-based non-parametric learner, such as with prototypical networks may be used, such that $\theta_p = \emptyset$. Metric-based

classifiers are more effective than probabilistic classifiers for novelty detection and have better training efficiency for large backbone networks than nested-loop training of optimization-based methods.

[0038] A model parameter θ is determined that maximizes the likelihood of observing a task τ. Thus the sample embedding may be $f_\theta(\mathbf{x}_i^*) = \mathbf{e}_i^* \in \mathbb{R}^d$, with the likelihood of the joint distribution $p_\theta(\mathbf{e}_i^*, y_i^*)$ being maximized on the observed data in $\mathcal{D}_\tau = \{\mathcal{D}_\tau^s, \mathcal{D}_\tau^q\}$, and where d is a hyper-parameter denoting a number of dimensions of the task representation. Each task τ may be considered as an instance, with a representation $\mathbf{v}_\tau \in \mathbb{R}^d$ in the embedding space. To model the unobserved mixture component, every task may be associated with a latent variable z_τ to indicate to which component it belongs.

[0039] With r possible components and a total number of samples $n = n_s + n_q$ in $\mathcal{D}_\tau$, the log-likelihood to maximize can be written by hierarchically factorizing on y_i^* and marginalizing out $\mathbf{v}_\tau$ and z_τ:

$$\ell(\mathcal{D}_\tau; \theta) = \frac{1}{n}\sum_{i=1}^{n} \log\left[p_\theta(\mathbf{e}_i^*|y_i^*)\left[\int_{\mathbf{v}_\tau} p(y_i^*|\mathbf{v}_\tau)\left[\sum_{z_\tau=1}^{r} p(\mathbf{v}_\tau|z_\tau)p(z_\tau)\right]d\mathbf{v}_\tau\right]\right]$$

where $p_\theta(\boldsymbol{e}_i^*|y_i^*)$ specifies the probability of sampling the embedding $\mathbf{e}_i^*$ from the class y_i^*, $p(y_i^*|\mathbf{v}_\tau)$ is the probability of sampling the class y_i^* for the task τ, and $p(\mathbf{v}_\tau|z_\tau)$ indicates the probability of generating a task τ from the mixture component z_τ. Thus $p(z_\tau)$ is a prior on the component z_τ, so that the task τ has a generative process $z_\tau \rightarrow \mathbf{v}_\tau \rightarrow y_i^* \rightarrow \mathbf{e}_i^*$.

[0040] The class-conditional distribution $p_\theta(\boldsymbol{e}_i^*|y_i^*)$, the task-conditional distribution $p(y_i^*|\mathbf{v}_\tau)$, and the mixture distribution of tasks defined by $\{p(\mathbf{v}_\tau|z_\tau), p(z_\tau)\}$ are not specified. To make the loss function ℓ optimizable, the generative process of tasks is

introduced. Because $\mathcal{D}_\tau^s$ and $\mathcal{D}_\tau^q$ follow the same distribution, the superscript * is ignored below for simplicity.

[0041] A Gaussian distribution may be used to model the embeddings $\mathbf{e}_i$ in each class. The mean $\boldsymbol{\mu}_{y_i}^c$ and the variance $\boldsymbol{\Sigma}_{y_i}^c$ of the class y_i produce $p_\theta(\mathbf{e}_i|y_i) = \mathcal{N}\left(\mathbf{e}_i|\boldsymbol{\mu}_{y_i}^c, \boldsymbol{\Sigma}_{y_i}^c\right)$. The samples in all of the classes of task τ make up a Gaussian mixture distribution, where $p(y_i)$ is the mixture probability of the class y_i. The term $p(y_i)$ may be factorized to be task specific, $p(y_i|\mathbf{v}_\tau)$, which resorts to another mixture distribution $\mathbf{v}_\tau$ of tasks and establishes a structure of hierarchical mixture.

[0042] However, the density function of the Gaussian distribution is log-concave with one global maximum. Given the mean and variance, maximizing its log-likelihood tends to collapse the prototypes $\boldsymbol{\mu}_{y_i}^c$ of all classes in τ, making them indistinguishable and impairing classification. Furthermore, given $\mathbf{v}_\tau$, this tends to sample classes with small $D_{\mathbf{v}_\tau}\left(\boldsymbol{\mu}_{y_i}^c\right)$, where $D_{\mathbf{v}_\tau}(\cdot)$ measures the Mahalanobis distance between a data point and the Gaussian distribution centered at $\mathbf{v}_\tau$. However, classes are often uniformly sampled from a domain without any prior on distances. Fitting the distance function with such uniform classes naively leads to an ill-posed learning problem with degenerate solutions.

[0043] As a result, $p(y_i|\mathbf{v}_\tau)$ may be defined as a parameterized density function with at least N global optima, so that it can distinguish the N different class prototypes of N-way tasks. The N equal optima also allow it to fit N classes uniformly sampled from a domain. To this end, $\boldsymbol{\mu}_{y_i}^c$ may be the surrogate embedding of the k^{th} class and a Gibbs distribution $\pi\left(\boldsymbol{\mu}_{y_i}^c|\mathbf{v}_\tau, \boldsymbol{w}\right)$ may be defined by v_τ and trainable parameters $\boldsymbol{w}$ with an energy function. Thus:

$$p_w(\boldsymbol{y_i} = \boldsymbol{k}|\mathrm{v}_\tau) = \pi(\boldsymbol{\mu}_k^c|\mathbf{v}_\tau, \boldsymbol{w}) = \frac{e^{-E_w(\boldsymbol{\mu}_k^c;\mathrm{v}_\tau)}}{\int_{\boldsymbol{\mu}_k^c} \boldsymbol{e}^{-E_w(\boldsymbol{\mu}_k^c;\mathrm{v}_\tau)}}$$

where $E_{\boldsymbol{w}}(\boldsymbol{\mu}_k^c; \mathbf{v}_\tau) = \min\left(\left\{\left\|\boldsymbol{\mu}_k^c - \mathbf{W}_j\mathbf{v}_\tau\right\|_2^2\right\}_{j=1}^N\right)$ is an energy function and the denominator is a normalizing constant with respect to $\boldsymbol{\mu}_k^c$, a partition function in an energy-based model, and $\boldsymbol{w} = \{\mathbf{W_1}, \ldots, \mathbf{W}_N\}$ are trainable parameters, with $\mathbf{W}_i \in \mathbb{R}^{d\times d}$. Given $\boldsymbol{w}$ and $\mathbf{v}_\tau$, there are N global maximums at $\boldsymbol{\mu}_k^c = \mathbf{W_1}\mathbf{v}_\tau, \ldots, \boldsymbol{\mu}_k^c = \mathbf{W}_N\mathbf{v}_\tau$.

[0044] The task distribution $p(\mathbf{v}_\tau)$ is factorized as a mixture of $p(\mathbf{v}_\tau|z_\tau = 1), \ldots, p(\mathbf{v}_\tau|z_\tau = r)$, weighted by their respective mixture probability $p(z_\tau)$. Thus $p(\mathbf{v}_\tau)$ may be specified as a Gaussian mixture distribution, with mean $\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}$ and variance $\boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}}$ of each component, such that $p(\mathbf{v}_\tau|z_\tau) = \mathcal{N}\left(\mathbf{v}_\tau\middle|\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}, \boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}}\right)$. The term $\mathbf{v}_\tau$ may be generated in two steps: drawing a latent task variable z_τ from a categorical distribution on $[p(z_\tau = 1), \ldots, p(z_\tau = r)]$, which can be $Uniform(r)$, and drawing $\mathbf{v}_\tau$ from $\mathcal{N}\left(\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}, \boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}}\right)$. The generative process of an N-way, K-shot, Q-query task τ can be summarized as:

[0045] 1. Draw $z_\tau \sim \mathrm{Categorical}([p(z_\tau = 1), \ldots, p(z_\tau = r)])$

[0046] 2. Draw a task embedding $\mathbf{v}_\tau \sim \mathcal{N}\left(\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}, \boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}}\right)$

[0047] 3. For $k = 1, \ldots, N$

[0048] (a) Draw a class prototype $\boldsymbol{\mu}_k^c \sim \pi(\boldsymbol{\mu}_k^c|\mathbf{v}_\tau, \boldsymbol{w})$ from Gibbs distribution

[0049] (b) For $i = 1, \ldots, K + Q$

[0050] i. Set $y_i = k$

[0051] ii. Draw $\mathbf{e}_i \sim \mathcal{N}\left(\mathbf{e}_i\middle|\boldsymbol{\mu}_{y_i}^{\mathrm{c}}, \boldsymbol{\Sigma}_{y_i}^{\mathrm{c}}\right)$

[0052] iii. If $i \leq K$:

[0053] (A) Allocate $(\mathbf{e}_i, y_i)$ to support set $\mathcal{D}_\tau^s$

[0054] iv. Else

[0055] (A) Allocate $(\mathbf{e}_i, y_i)$ to query set $\mathcal{D}_\tau^q$

[0056] To reduce complexity, an isotropic Gaussian with tied variance may be used for class distributions, where $\boldsymbol{\Sigma}_1^c = \cdots = \boldsymbol{\Sigma}_N^c = \sigma^2\mathbf{I}$, with $\mathbf{I}$ being the identity matrix and σ being a hyperparameter. For task distributions, the variances can be automatically inferred.

[0057] Substituting $p_\theta(\mathbf{e}_i|y_i) = \mathcal{N}(\mathbf{e}_i|\boldsymbol{\mu}_{y_i}^c, \sigma\mathbf{I})$, $p_w(y_i|\mathbf{v}_\tau) = \pi(\boldsymbol{\mu}_{y_i}^c|\mathbf{v}_\tau, \boldsymbol{w})(y_i = k)$, $p(\mathbf{v}_\tau|z_\tau) = \mathcal{N}(\mathbf{v}_\tau|\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}, \boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}})$, and $p(z_\tau) = Uniform(r)$, whose probabilities are specified and parameterized, a loss function $\ell(\mathcal{D}_\tau; \theta, w)$ is determined. The class means $\boldsymbol{\mu}_{y_i}^c$, task means $\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}$, and variances $\boldsymbol{\Sigma}_{z_\tau}^{\mathrm{t}}$ are inferred as described below.

[0058] Directly optimizing $\ell(\mathcal{D}_\tau; \theta, w)$ is challenging, because the exact posterior inference is intractable due to the integration over $\mathbf{v}_\tau$. Variational methods may be used to solve it, introducing an approximated posterior $q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)$, which is defined by an inference network ϕ and implies that $\mathbf{v}_\tau$ may be inferred from its observed support step $\mathcal{D}_\tau^s$. The query set $\mathcal{D}_\tau^q$ is not included because it is not available during model testing. A lower bound may be expressed as:

$$\ell_L(\mathcal{D}_\tau; \theta, w) = \frac{1}{N}\sum_{i=1}^{n}\Bigg(\log p_{\theta,w}(\mathbf{e}_i|y_i) + \mathbb{E}_{\mathbf{v}_\tau \sim q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)}\left[\log p_w(y_i|\mathbf{v}_\tau) + \log\sum_{z_\tau=1}^{r} p(\mathbf{v}_\tau|z_\tau)p(z_\tau)\right]\Bigg) + H\left(q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)\right)$$

where $H\left(q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)\right) = -\int_{\mathbf{v}_\tau} q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)\log q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)\, d\mathbf{v}_\tau$ is the entropy function. This formulation estimates the expectation $\mathbb{E}$ by sampling $\mathbf{v}_\tau$ from $q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)$ instead of the integration to facilitate computation.

[0059] The function $q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^s)$ may be defined as a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{z_\tau}^a, \bar{\sigma}^2\mathbf{I})$, where $\boldsymbol{\mu}_{z_\tau}^a$ is the output of the inference network that approximates $\boldsymbol{\mu}_{z_\tau}^{\mathrm{t}}$

and $\bar{\sigma}$ is a hyperparameter for the corresponding variance. The inference network may be built on a base model $f_\theta(\cdot)$ with two non-parametric aggregation functions, thus $\phi = \theta$. The first function aggregates class-wise embeddings to prototypes $\boldsymbol{\mu}^c_{y_i}$. The second aggregates prototypes to $\boldsymbol{\mu}^a_{z_\tau}$. During model training, reparameterization may be used to sample $\mathbf{v}_\tau$ from $\mathcal{N}(\boldsymbol{\mu}^a_{z_\tau}, \bar{\sigma}^2\mathbf{I})$. Meanwhile $H\left(q_\phi(\mathbf{v}_\tau|\mathcal{D}^s_\tau)\right)$ becomes a constant due to $\bar{\sigma}^2$ being a constant.

[0060] In some cases, different class means $\boldsymbol{\mu}^c_1, \ldots, \boldsymbol{\mu}^c_N$ in a task τ may collide, drawing all sample embeddings to the same spot. To avoid trivial solutions and to improve the stability of optimization, negative sampling may be used:

$$\ell_{neg}(\mathcal{D}_\tau; y_i, \theta, \boldsymbol{w}) = -\log \mathbb{E}_{\mathbf{e}_j \sim \mathcal{D}_\tau}\left[e^{-\frac{\left\|\mathbf{e}_j - \mu^c_{y_i}\right\|_2^2}{2\sigma^{2d}}}\right]$$

where $\mathbf{e}_j$ is a negative sample embedding rom any class in the support set and where $\boldsymbol{\mu}^c_{y_i}$ is the mean of the positive class. It is beneficial to integrate ℓ with the likelihood loss function during training:

$$\ell_L(\mathcal{D}_\tau; \theta, \boldsymbol{w}) + \frac{1}{n}\sum_{i=1}^{n} \ell_{neg}(\mathcal{D}_\tau; y_i, \theta, \boldsymbol{w})$$

This not only serves as a robust training lost, but also helps to solve challenges relating to the partition function.

[0061] In particular, the term $p_w(y_i|\mathbf{v}_\tau)$ involves computing a partition function, which may be intractable due to the integration over all possible $\boldsymbol{\mu}^c_k$. An upper bound may be imposed on the partition function:

$$\int_{\boldsymbol{\mu}_k} e^{-E_w(\boldsymbol{\mu}^c_k; \mathbf{v}_\tau)} d\boldsymbol{\mu}^c_k \leq N\sqrt{2^{d-1}\pi^d}$$

a constant value with a specific N. This upper bound shows the partition function approximates $N\sqrt{2^{d-1}\pi^d}$ when all pairs of the global maximums are far apart. By replacing he partition function with $N\sqrt{2^{d-1}\pi^d}$, a lower bound for $p_w(y_i|\mathbf{v}_\tau)$ is obtained, relaxing the lower bound of ℓ_L.

[0062] The tightness of the relaxed bound is controllable. Among the N global maxima $\mathbf{W}_1\mathbf{v}_\tau, \ldots, \mathbf{W}_N\mathbf{v}_\tau$, the terms $\mathbf{W}_h\mathbf{v}_\tau$ and $\mathbf{W}_l\mathbf{v}_\tau$ are the terms with the smallest Euclidean distance D_{hl}, where $1 \leq h$ and $l \leq N$. Thus:

$$\lim_{D_{hl}\to\infty} \int_{\boldsymbol{\mu}_k} e^{-E_w(\boldsymbol{\mu}_k^c;\mathbf{v}_\tau)} d\boldsymbol{\mu}_k^c = N\sqrt{2^{d-1}\pi^d}$$

The partition function therefore approximates $N\sqrt{2^{d-1}\pi^d}$ when all pairs of the global maxima are far apart. When maximizing the likelihood, $\mathbf{W}_1\mathbf{v}_\tau, \ldots, \mathbf{W}_N\mathbf{v}_\tau$ is fit to different class prototypes $\boldsymbol{\mu}_1^c, \ldots, \boldsymbol{\mu}_N^c$ in N-way tasks. Because ℓ_{neg} tends to maximize the distances between different prototypes through negative samples, maximizing the joint loss tends to separate $\mathbf{W}_1\mathbf{v}_\tau, \ldots, \mathbf{W}_N\mathbf{v}_\tau$, thus tightening the relaxed bound after using $N\sqrt{2^{d-1}\pi^d}$.

[0063] The mixture distribution $p(z_\tau)$ is estimated in the third term of ℓ_L. Similar to optimizing Gaussian mixture models, $p(z_\tau)$ may be inferred and the model parameters $\{\theta, w\}$ may be solved through an expectation-minimization process. In an expectation step, $p(z_\tau)$ may be inferred when fixing model parameters. In a maximization step, $\{\theta, w\}$ may be solved by fixing $p(z_\tau)$ and optimizing the combined loss with stochastic gradient descent.

[0064] In fine-tuning 206, the model may be adapted to new environmental information. Given a new N-way task τ' from the meta-test set $\mathcal{D}^{te}$, the corresponding support set $\mathcal{D}_{\tau'}^s$ is fed into the network to generate class prototypes $\boldsymbol{\mu}_1^c, \ldots \boldsymbol{\mu}_N^c$ and distribution $q_\phi(\mathbf{v}_{\tau'}|\mathcal{D}_{\tau'}^s)$, from which average task embedding $\mathbf{v}_{\tau'} = \boldsymbol{\mu}_{z_{\tau'}}^a$ may be

drawn. The inference network is the base model $f_\theta(\cdot)$ with class-pooling and task-pooling layers, and $\phi = \theta$.

[0065] The task embedding $\mathbf{v}_{\tau'}$ is projected to $\mathbf{W}_1\mathbf{v}_{\tau'}, \ldots, \mathbf{W}_N\mathbf{v}_{\tau'}$, which represent the N optimal choices of class prototypes for task τ' as learned by the Gibbs distribution from training the tasks. They are used to adapt $\boldsymbol{\mu}_1^c, \ldots \boldsymbol{\mu}_N^c$ so that the adapted prototypes are drawn toward the closest classes from the mixture component that task τ' belongs to. The adaptation is performed by selecting the closest optimum for each prototype:

$$\bar{\boldsymbol{\mu}}_j^c = \alpha\boldsymbol{\mu}_j^c + (1-\alpha)\mathbf{W}_{l^*}\mathbf{v}_{\tau'}$$

where $l^* = \arg min_{1\le l\le N} D\left(\boldsymbol{\mu}_j^c, \mathbf{W}_l\mathbf{v}_{\tau'}\right)$ using Euclidean distance $D(\cdot,\cdot)$ and hyperparameter α. It is determined whether τ' is a novelty by computing the likelihood of $\mathbf{v}_{\tau'}$ in a pre-fitted Gaussian mixture model (GMM) on the embeddings $\mathbf{v}_{\tau'}$ of the training tasks in $\mathcal{D}^{tr}$ and classification is performed on each sample $\mathbf{x}_s'$ in the query set $\mathcal{D}_{\tau'}^q$ using adapted prototypes by:

$$p(y_i' = j'|x_i') = \frac{e^{-D\left(f_\theta(\mathbf{x}_i'), \bar{\boldsymbol{\mu}}_{j'}^c\right)}}{\sum_{j=1}^N e^{-D\left(f_\theta(\mathbf{x}_i'), \bar{\boldsymbol{\mu}}_j^c\right)}}$$

[0066] Referring now to FIG. 3, detail on a task-generation meta-learning (TGML) model 300 is shown. The TGML model 300 allows heterogeneous data distributions of different tasks in the experimental environment, so that different tasks can have different data distributions. The TGML model 300 further incorporates the configuration values of the environmental or user conditions as a task-level meta-feature for improving task representation learning. Models can be quickly adapted to a new task, with automatic detection of out-of-distribution tasks, which can serve as an indication of whether to trust the results from the data. While the TGML model 300 is described herein as pertaining specifically to processing signal data from optical communications networks, it should be understood that the present principles may be

applied to other types of data, such as images, text, video, etc., with an appropriate choice of data encoder.

[0067] A time series encoder 302 accepts signal data from a task and transforms samples of the multivariate time series data to a dense vector representation, for example using a long-short term memory (LSTM) neural network:

$$\mathbf{e} = LSTM([\mathbf{x}_1, \ldots, \mathbf{x}_T])$$

where $\mathbf{x}_1, \ldots \mathbf{x}_T$ represent the T measurements within the sample.

[0068] A class prototype encoder 304 learns a prototype vector that represents each class in the task. The input data may include $\{(\mathbf{e}_1, y_1), \ldots, (\mathbf{e}_n, y_n)\}$, where y_i is the label of the i^{th} sample having a value within $[1, \ldots, N]$ for an N-way classification. The class prototype encoder computes a prototype vector $\mathbf{c}_j$ for each class j, with values $1 \leq j \leq N$:

$$\mathbf{c}_j = \frac{1}{n_k} \sum_{[(\mathbf{v}_i, y_i)]_{y_i = j}} MLP(\mathbf{e}_i)$$

where $MLP(\cdot)$ represents a multilayer perceptron (MLP) neural network. This summation obtains the averaged embedding of all the embeddings of instances in the same category j.

[0069] An inference network 306 learns a task embedding by integrating the information in task-level features and time series samples and then labels the task. As described in greater detail below, the inference network 306 includes a task-level feature encoder that encodes configuration data for the environment, a task instance encoder that aggregates class prototypes learned by the class prototype encoder 304, and a task embedder that aggregates the representations from the task-level feature encoder and the task instance encoder.

[0070] TGML training 308 performs a maximum likelihood. As will be described in greater detail below, the TGML training 308 optimizes model parameters of the encoding networks of the time series encoder 302, the class prototype encoder 304, and the inference network 306 and a set of distribution parameters $\{\mu_z\}$.

[0071] TGML testing 310 tests the trained model on new tasks. A class prototype adapter takes an average of support set samples of each class with its closest class prototype from the training set as the adapted prototype of the support set class. Using the adapted class prototypes, new time series samples in a query set can be classified using a distance based classifier.

[0072] Referring now to FIG. 4, additional detail is shown in the inference network 306. A task-level feature encoder 402 may use an MLP neural network to encode configuration data from the environmental setup of the optical network. The configuration data may encapsulate useful information regarding the relationships between different tasks. Although an MLP encoder is specifically contemplated for the task-level features, it should be understood that any other appropriate encoder structure may be used instead, such as a text encoder for encoding textual configuration information.

[0073] A task instance encoder 404 aggregates the class prototypes learned by the class prototype encoder 304, which encodes the information of the time series data and labels in the task. The aggregation may be performed as:

$$\mathbf{v} = \frac{1}{N}\sum_{j=1}^{N} MLP(\mathbf{c}_j)$$

The task embedder 406 aggregates the representations from the task-level feature encoder 402 and the task instance encoder 404 by, e.g., concatenating the two vector representations and feeding them to an MLP.

[0074] Referring now to FIG. 5, additional detail is shown in the TGML training 308. A task embedding modulator 502 adjusts the task embedding $\mathbf{v}$ by projecting it to different areas by multiplying respective modulating vectors $[\mathbf{W}_1, \ldots, \mathbf{W}_N]$ as discussed above to generate $[\mathbf{W}_1\mathbf{v}, \ldots, \mathbf{W}_N\mathbf{v}]$. This addresses the tendency of the generative process to generate classes that are similar to one another. A task-class distance function 504 avoids a trivial solution where tasks are sampled randomly and fitted to a model that locates the task embeddings at the center of a task cluster, whereas class means are on the hypersphere of the Gaussian distribution and have equal distance to a task embedding. The distance function may be expressed as:

$$D(\boldsymbol{\mu}_y, \mathbf{v}) = \max\left(\{\boldsymbol{\mu}_y - \boldsymbol{W}_i\boldsymbol{v}\}_{i=1}^{N}\right)$$

which has global minima and helps alleviate the problem because the randomly sampled classes tend to fit different global minima. The term $\boldsymbol{\mu}_y$ denotes class prototypes of class y. This allows the model to fit randomly sampled classes.

[0075] A time series density estimator 506 determines the probability of time series samples $p(\mathbf{e}|y)$ using their embeddings, given their class labels. Gaussian mixtures are used for time series samples, so the probability is:

$$p(\mathbf{e}|y) = p_N(\mathbf{e}|\boldsymbol{\mu}_y, \Sigma_y)$$

where $p_N(\cdot)$ represents the density function of the Gaussian distribution, $\boldsymbol{\mu}_y$ is the class prototype of the class represented by y (as output by class prototype encoder 304), and Σ_y is the covariance matrix.

[0076] A task density estimator 508 computes the probability of a task $p(\mathbf{v}|y)$ using the embeddings from the task embedder 406. Again using the Gaussian mixture for the task embeddings, the probability is computed as:

$$p(\mathbf{v}|y) = p_N(\mathbf{v}|\boldsymbol{\mu}_z, \Sigma_z)$$

where $\boldsymbol{\mu}_z$ is the cluster center of the cluster represented by z, which are model parameters, and Σ_z is the covariance matrix. If the density of a testing task is below a predetermined threshold, it may be identified as an out-of-distribution task. An out-of-distribution task may be reported to an administrator, though classification may still be performed with reduced confidence.

[0077] Using the outputs of the task-class distance function 504, the time series density estimator 506, and the task density estimator 508, the likelihood function 510 of the generative model may be expressed as:

$$\begin{aligned}\log\big(p(x,y)\big) &= \log\left(p(x|y)\int_v p(y|v)p(v)dv\right)\\ &\geq log\, p(x|y) + \mathbb{E}_{q(v|f_{task},x_{sup},y)}[log\, p(y|v)]\\ &+ \mathbb{E}_{q(v|f_{task},x_{sup},y),\, q(z|v)}[log\, p(v|z)] - \mathbb{E}_{q(v|y)}\big[KL\big(q(z|v)||p(z)\big)\big]\\ &- H\big(q(v|f_{task},x_{sup},y)\big)\end{aligned}$$

which is a lower bound of the likelihood of the observed samples and labels $p(x,y)$ in a task. The lower bound $\ell_L(\mathcal{D}_\tau;\theta,w)$, described above, may be used instead. An optimizer 512 optimizes the model parameters in the encoding networks and the distribution-related parameters $\{\boldsymbol{\mu}_z\}$. Any appropriate optimization, such as an Adam optimizer, may be used.

[0078] Referring now to FIG. 6, a method of training a TGML model is shown. The training starts with training tasks $\mathcal{D}_{train}$, with each task having time series samples and labels. A task-level configuration file is further provided. Time series samples are encoded 602 and class prototypes are determined 604. Block 606 encodes the task. As noted above, block 608 computes a likelihood function, which block 610 optimizes according to the loss function described above.

[0079] Referring now to FIG. 7, a method of classifying the behavior of a new system is shown. Block 702 encodes the test task. In the context of an optical network, this may include taking multivariate time series measurements of received signals, after polarization decomposition and after DSP, and encoding the time series as described above.

[0080] Block 704 estimates the task density. Using this density information, block 706 determines that the block is not out of distribution. This indicates that the state of the system is within the range that the trained model can handle. Block 708 then uses the model to classify the time series samples, for example to generate SNR information relating to the optical communications network.

[0081] Based on the outcome of the classification, a responsive action may be automatically performed 710 to improve the SNR or data rate. For example, if the SNR is low (e.g., below a threshold value that corresponds to accurate decoding of the signals), parameters of the signal may be altered to improve the SNR. If the SNR is high (e.g., above a threshold value that indicates the signal can be easily decoded), the parameters of the signal may be altered to improve the data rate. This adjustment may include changing modulation or encoding parameters of the signal and/or may include changing physical parameters of the optical network, such as altering environmental conditions of the optical networking equipment and correcting any malfunctions in the equipment itself.

[0082] Referring now to FIG. 8, an exemplary computing device 800 is shown, in accordance with an embodiment of the present invention. The computing device 800 is configured to perform model training and network management.

[0083] The computing device 800 may be embodied as any type of computation or computer device capable of performing the functions described herein, including,

without limitation, a computer, a server, a rack based server, a blade server, a workstation, a desktop computer, a laptop computer, a notebook computer, a tablet computer, a mobile computing device, a wearable computing device, a network appliance, a web appliance, a distributed computing system, a processor- based system, and/or a consumer electronic device. Additionally or alternatively, the computing device 800 may be embodied as one or more compute sleds, memory sleds, or other racks, sleds, computing chassis, or other components of a physically disaggregated computing device.

[0084] As shown in FIG. 8, the computing device 800 illustratively includes the processor 810, an input/output subsystem 820, a memory 830, a data storage device 840, and a communication subsystem 850, and/or other components and devices commonly found in a server or similar computing device. The computing device 800 may include other or additional components, such as those commonly found in a server computer (e.g., various input/output devices), in other embodiments. Additionally, in some embodiments, one or more of the illustrative components may be incorporated in, or otherwise form a portion of, another component. For example, the memory 830, or portions thereof, may be incorporated in the processor 810 in some embodiments.

[0085] The processor 810 may be embodied as any type of processor capable of performing the functions described herein. The processor 810 may be embodied as a single processor, multiple processors, a Central Processing Unit(s) (CPU(s)), a Graphics Processing Unit(s) (GPU(s)), a single or multi-core processor(s), a digital signal processor(s), a microcontroller(s), or other processor(s) or processing/controlling circuit(s).

[0086] The memory 830 may be embodied as any type of volatile or non-volatile memory or data storage capable of performing the functions described herein. In

operation, the memory 830 may store various data and software used during operation of the computing device 800, such as operating systems, applications, programs, libraries, and drivers. The memory 830 is communicatively coupled to the processor 810 via the I/O subsystem 820, which may be embodied as circuitry and/or components to facilitate input/output operations with the processor 810, the memory 830, and other components of the computing device 800. For example, the I/O subsystem 820 may be embodied as, or otherwise include, memory controller hubs, input/output control hubs, platform controller hubs, integrated control circuitry, firmware devices, communication links (e.g., point-to-point links, bus links, wires, cables, light guides, printed circuit board traces, etc.), and/or other components and subsystems to facilitate the input/output operations. In some embodiments, the I/O subsystem 820 may form a portion of a system-on-a-chip (SOC) and be incorporated, along with the processor 810, the memory 830, and other components of the computing device 800, on a single integrated circuit chip.

[0087] The data storage device 840 may be embodied as any type of device or devices configured for short-term or long-term storage of data such as, for example, memory devices and circuits, memory cards, hard disk drives, solid state drives, or other data storage devices. The data storage device 840 can store program code 840A for performing training of the TGML model, 840B for testing a network's conditions with the TGML model, and/or 840C for network management. Any or all of these program code blocks may be included in a given computing system. The communication subsystem 850 of the computing device 800 may be embodied as any network interface controller or other communication circuit, device, or collection thereof, capable of enabling communications between the computing device 800 and other remote devices over a network. The communication subsystem 850 may be configured to use any one

or more communication technology (e.g., wired or wireless communications) and associated protocols (e.g., Ethernet, InfiniBand®, Bluetooth®, Wi-Fi®, WiMAX, etc.) to effect such communication.

[0088] As shown, the computing device 800 may also include one or more peripheral devices 860. The peripheral devices 860 may include any number of additional input/output devices, interface devices, and/or other peripheral devices. For example, in some embodiments, the peripheral devices 860 may include a display, touch screen, graphics circuitry, keyboard, mouse, speaker system, microphone, network interface, and/or other input/output devices, interface devices, and/or peripheral devices.

[0089] Of course, the computing device 800 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 sensors, input devices, and/or output devices can be included in computing device 800, 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. These and other variations of the processing system 800 are readily contemplated by one of ordinary skill in the art given the teachings of the present invention provided herein.

[0090] Referring now to FIGs. 9 and 10, exemplary neural network architectures are shown, which may be used to implement parts of the present models. A neural network is a generalized system that improves its functioning and accuracy through exposure to additional empirical data. The neural network becomes trained by exposure to the empirical data. During training, the neural network stores and adjusts a plurality of weights that are applied to the incoming empirical data. By applying the adjusted

weights to the data, the data can be identified as belonging to a particular predefined class from a set of classes or a probability that the inputted data belongs to each of the classes can be outputted.

[0091] The empirical data, also known as training data, from a set of examples can be formatted as a string of values and fed into the input of the neural network. Each example may be associated with a known result or output. Each example can be represented as a pair, (x, y), where x represents the input data and y represents the known output. The input data may include a variety of different data types, and may include multiple distinct values. The network can have one input node for each value making up the example's input data, and a separate weight can be applied to each input value. The input data can, for example, be formatted as a vector, an array, or a string depending on the architecture of the neural network being constructed and trained.

[0092] The neural network "learns" by comparing the neural network output generated from the input data to the known values of the examples, and adjusting the stored weights to minimize the differences between the output values and the known values. The adjustments may be made to the stored weights through back propagation, where the effect of the weights on the output values may be determined by calculating the mathematical gradient and adjusting the weights in a manner that shifts the output towards a minimum difference. This optimization, referred to as a gradient descent approach, is a non-limiting example of how training may be performed. A subset of examples with known values that were not used for training can be used to test and validate the accuracy of the neural network.

[0093] During operation, the trained neural network can be used on new data that was not previously used in training or validation through generalization. The adjusted weights of the neural network can be applied to the new data, where the weights

estimate a function developed from the training examples. The parameters of the estimated function which are captured by the weights are based on statistical inference.

[0094] In layered neural networks, nodes are arranged in the form of layers. An exemplary simple neural network has an input layer 920 of source nodes 922, and a single computation layer 930 having one or more computation nodes 932 that also act as output nodes, where there is a single computation node 932 for each possible category into which the input example could be classified. An input layer 920 can have a number of source nodes 922 equal to the number of data values 912 in the input data 910. The data values 912 in the input data 910 can be represented as a column vector. Each computation node 932 in the computation layer 930 generates a linear combination of weighted values from the input data 910 fed into input nodes 920, and applies a non-linear activation function that is differentiable to the sum. The exemplary simple neural network can perform classification on linearly separable examples (e.g., patterns).

[0095] A deep neural network, such as a multilayer perceptron, can have an input layer 920 of source nodes 922, one or more computation layer(s) 930 having one or more computation nodes 932, and an output layer 940, where there is a single output node 942 for each possible category into which the input example could be classified. An input layer 920 can have a number of source nodes 922 equal to the number of data values 912 in the input data 910. The computation nodes 932 in the computation layer(s) 930 can also be referred to as hidden layers, because they are between the source nodes 922 and output node(s) 942 and are not directly observed. Each node 932, 942 in a computation layer generates a linear combination of weighted values from the values output from the nodes in a previous layer, and applies a non-linear activation function that is differentiable over the range of the linear combination. The weights

applied to the value from each previous node can be denoted, for example, by w_1, w_2, ... w_{n-1}, w_n. The output layer provides the overall response of the network to the input data. A deep neural network can be fully connected, where each node in a computational layer is connected to all other nodes in the previous layer, or may have other configurations of connections between layers. If links between nodes are missing, the network is referred to as partially connected.

[0096] Training a deep neural network can involve two phases, a forward phase where the weights of each node are fixed and the input propagates through the network, and a backwards phase where an error value is propagated backwards through the network and weight values are updated.

[0097] The computation nodes 932 in the one or more computation (hidden) layer(s) 930 perform a nonlinear transformation on the input data 912 that generates a feature space. The classes or categories may be more easily separated in the feature space than in the original data space.

[0098] 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.

[0099] 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.

[0100] 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.

[0101] 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.

[0102] 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.

[0103] 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.).

[0104] 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.

[0105] 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).

[0106] These and other variations of a hardware processor subsystem are also contemplated in accordance with embodiments of the present invention.

[0107] 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.

[0108] 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.

[0109] 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 for training a model, comprising:

determining (604) class prototypes of time series samples from a training dataset;

encoding (606) a task corresponding to the time series samples using the class prototypes and a task-level configuration;

determining (608) a likelihood value based on outputs of a time series density model, a task-class distance from a task embedding model, and a task density model; and

adjusting (610) parameters of the time series density model, the task embedding model, and the task density model responsive to the likelihood value.

2. The method of claim 1, further comprising encoding the time series samples into a vector representation before determining the class prototypes.

3. The method of claim 1, wherein encoding the task includes encoding task level features that include the task-level configuration and encoding task instance features by aggregating the class prototype.

4. The method of claim 1, wherein the task density model determines a task density as a probability of a task given a cluster center and a covariance matrix of a cluster.

5. The method of claim 1, wherein the time series density model determines a time series density as a probability of a time series sample based on a class prototype and a covariance matrix of a class.

6. The method of claim 1, wherein the task-class distance is determined as a maximum of a distance between a prototype and a set of optimal class prototypes.

7. The method of claim 1, wherein the likelihood value has a lower bound:

$$\ell_L(\mathcal{D}_\tau;\theta,w) = \frac{1}{N}\sum_{i=1}^{n}\Bigg(\log p_{\theta,w}(\mathbf{e}_i|y_i) + \mathbb{E}_{\mathbf{v}_\tau\sim q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^S)}\left[\log p_w(y_i|\mathbf{v}_\tau) + \log\sum_{z_\tau=1}^{r} p(\mathbf{v}_\tau|z_\tau)p(z_\tau)\right]\Bigg) + H\left(q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^S)\right)$$

where $H(\cdot)$ is an entropy function, N is a number of classes, $p_{\theta,w}(\mathbf{e}_i|\mathbf{y}_i)$ is a probability of an embedding $\mathbf{e}_i$ given a sample y_i with model parameter θ and trainable parameters w, $\mathbb{E}$ is an expectation function, $\mathbf{v}_\tau$ is an embedded representation of task τ, $\mathcal{D}_\tau^S$ is a set of training support samples, $q_\phi(\cdot)$ is an approximated posterior with inference network ϕ, z_τ is a latent task variable, and r is a hyperparameter of a number of components in a mixture distribution.

8. The method of claim 1, wherein the task-level configuration includes configuration parameters relating to an environment of an optical communications network and wherein the time series samples include a measurement of an optical communications signal.

9. The method of claim 8, further comprising determining a new task-level configuration that reflects configuration parameters relating to a new environment, to adapt to the new environment.

10. The method of claim 8, wherein the time series samples include one or more of data obtained after polarization decomposition and data obtained after digital signal processing.

11. A system for training a model, comprising:

a hardware processor (810); and

a memory (840) that stores a computer program which, when executed by the hardware processor, causes the hardware processor to:

determine (604) class prototypes of time series samples from a training dataset;

encode (606) a task corresponding to the time series samples using the class prototypes and a task-level configuration;

determine (608) a likelihood value based on outputs of a time series density model, a task-class distance from a task embedding model, and a task density model; and

adjust (610) parameters of the time series density model, the task embedding model, and the task density model responsive to the likelihood value.

12. The system of claim 11, wherein the computer program further causes the hardware processor to encode the time series samples into a vector representation before determining the class prototypes.

13. The system of claim 11, wherein the computer program further causes the hardware processor to encode task level features that include the task-level configuration and to encode task instance features by aggregating the class prototype.

14. The system of claim 11, wherein the task density model determines a task density as a probability of a task given a cluster center and a covariance matrix of a cluster.

15. The system of claim 11, wherein the time series density model determines a time series density as a probability of a time series sample based on a class prototype and a covariance matrix of a class.

16. The system of claim 11, wherein the task-class distance is determined as a maximum of a distance between a prototype and a set of optimal class prototypes.

17. The system of claim 11, wherein the likelihood value has a lower bound:

$$\ell_L(\mathcal{D}_\tau;\theta,w) = \frac{1}{N}\sum_{i=1}^{n}\Bigg(\log p_{\theta,w}(\mathbf{e}_i|y_i) + \mathbb{E}_{\mathbf{v}_\tau\sim q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^S)}\left[\log p_w(y_i|\mathbf{v}_\tau) + \log\sum_{z_\tau=1}^{r} p(\mathbf{v}_\tau|z_\tau)p(z_\tau)\right]\Bigg) + H\left(q_\phi(\mathbf{v}_\tau|\mathcal{D}_\tau^S)\right)$$

where $H(\cdot)$ is an entropy function, N is a number of classes, $p_{\theta,w}(\mathbf{e}_i|\mathbf{y}_i)$ is a probability of an embedding $\mathbf{e}_i$ given a sample y_i with model parameter θ and trainable parameters w, $\mathbb{E}$ is an expectation function, $\mathbf{v}_\tau$ is an embedded representation of task τ, $\mathcal{D}_\tau^S$ is a set of training support samples, $q_\phi(\cdot)$ is an approximated posterior with inference network ϕ, z_τ is a latent task variable, and r is a hyperparameter of a number of components in a mixture distribution.

18. The system of claim 11, wherein the task-level configuration includes configuration parameters relating to an environment of an optical communications network and wherein the time series samples include a measurement of an optical communications signal.

19. The system of claim 18, wherein the computer program further causes the hardware processor to determine a new task-level configuration that reflects configuration parameters relating to a new environment, to adapt to the new environment.

20. The system of claim 18, wherein the time series samples include one or more of data obtained after polarization decomposition and data obtained after digital signal processing.

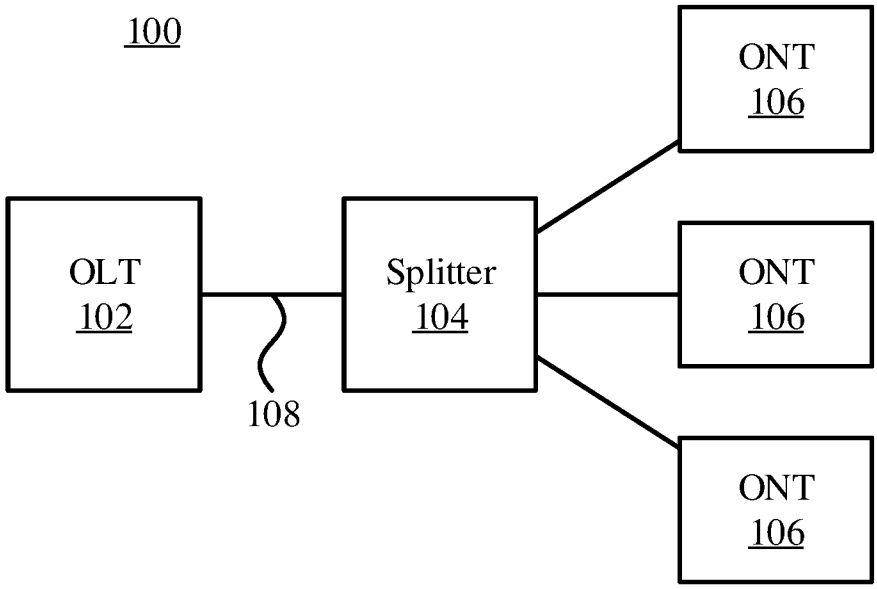
100
ONT
106
OLT
102
Splitter
104
ONT
106
108
ONT
106

FIG. 1

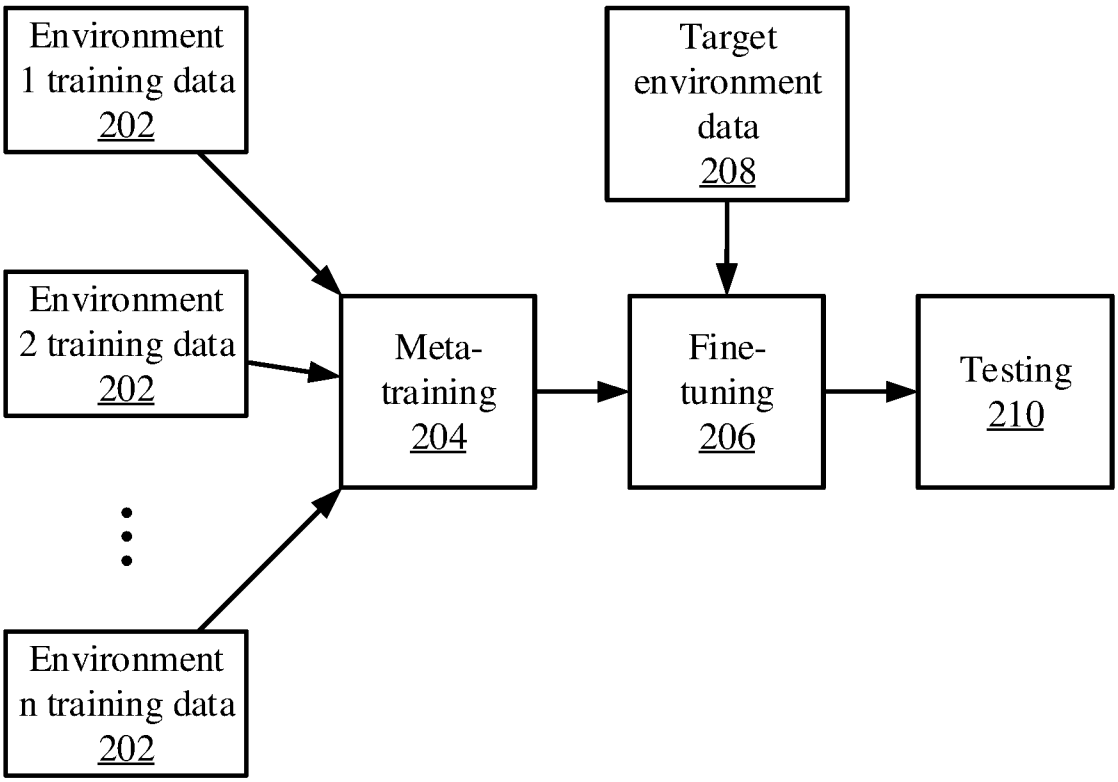
Environment
1 training data
202
Target
environment
data
208
Environment
2 training data
202
Meta-
training
204
Fine-
tuning
206
Testing
210
Environment
n training data
202

FIG. 2

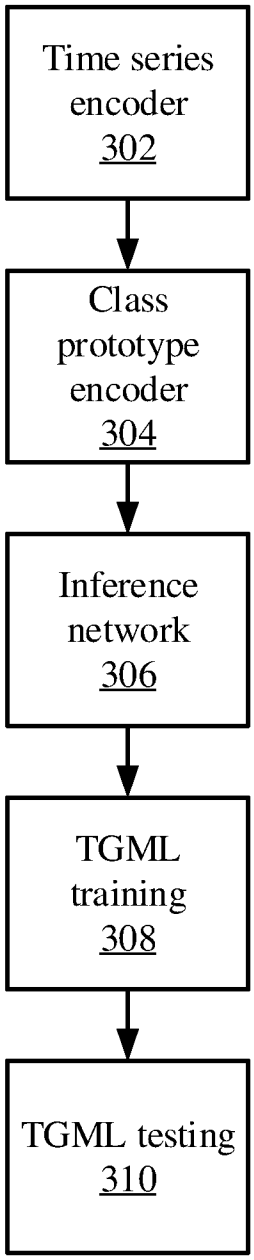
Time series encoder 302
Class prototype encoder 304
Inference network 306
TGML training 308
TGML testing 310

FIG. 3

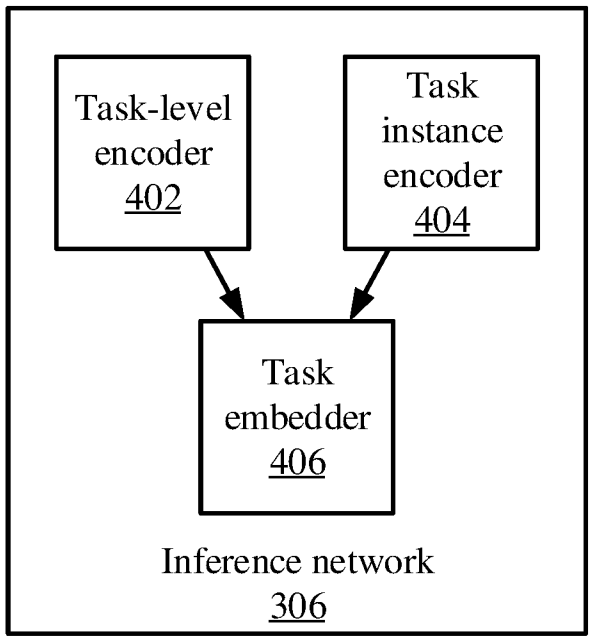
Task-level encoder 402
Task instance encoder 404
Task embedder 406
Inference network 306

FIG. 4

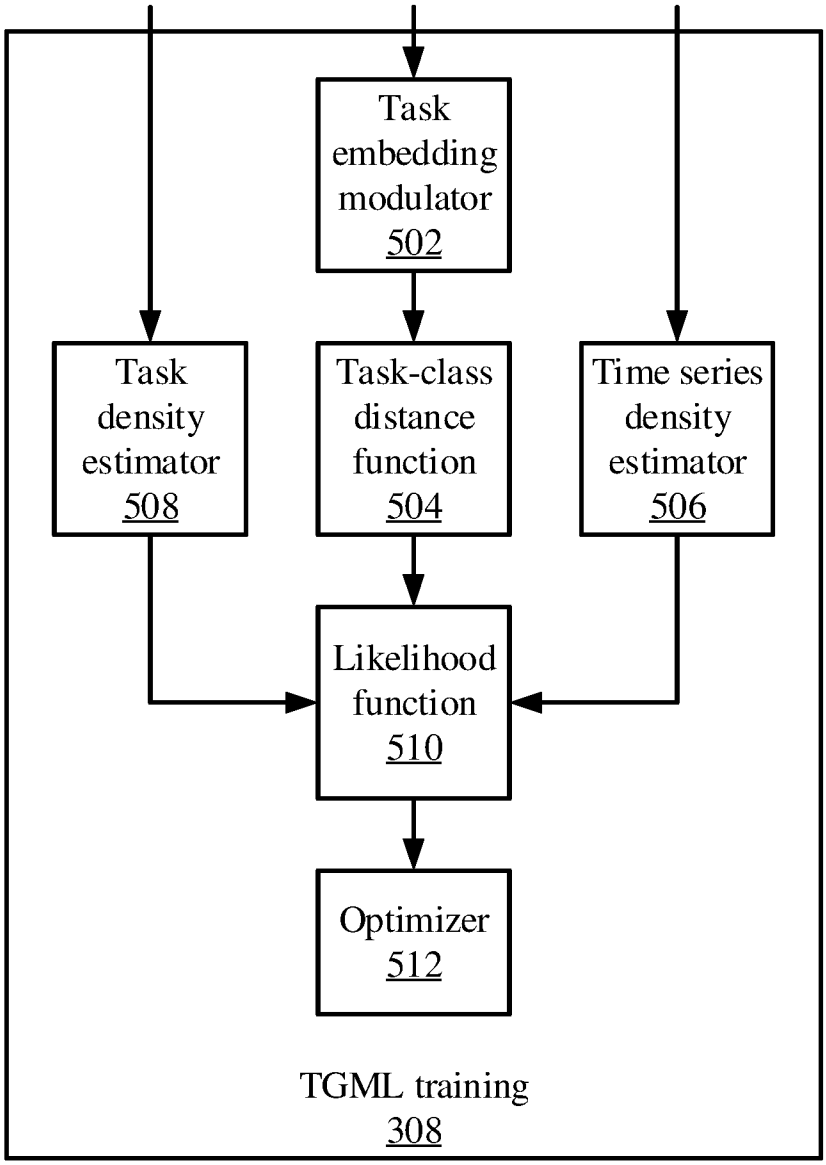
Task embedding modulator 502
Task density estimator 508
Task-class distance function 504
Time series density estimator 506
Likelihood function 510
Optimizer 512
TGML training 308

FIG. 5

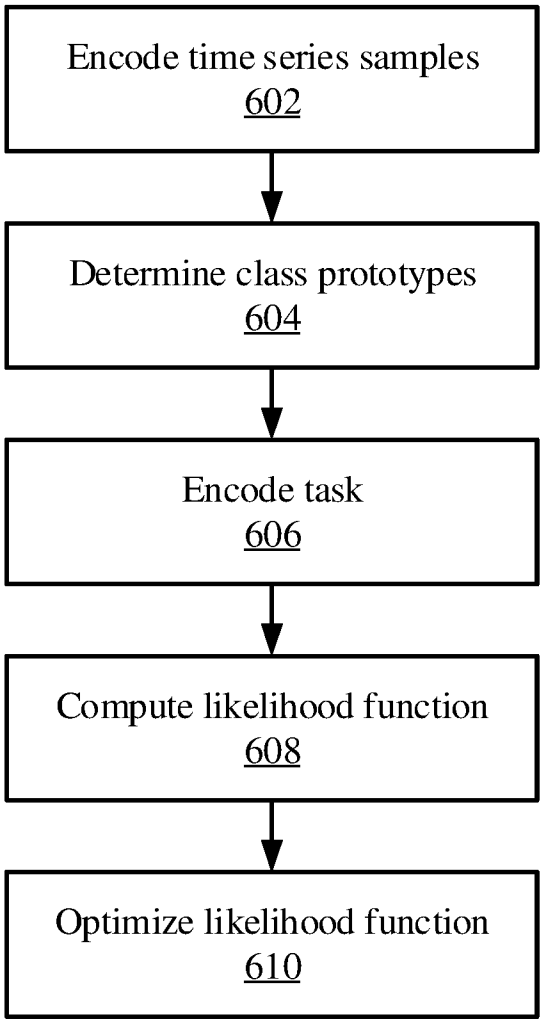
Encode time series samples
602
Determine class prototypes
604
Encode task
606
Compute likelihood function
608
Optimize likelihood function
610

FIG. 6

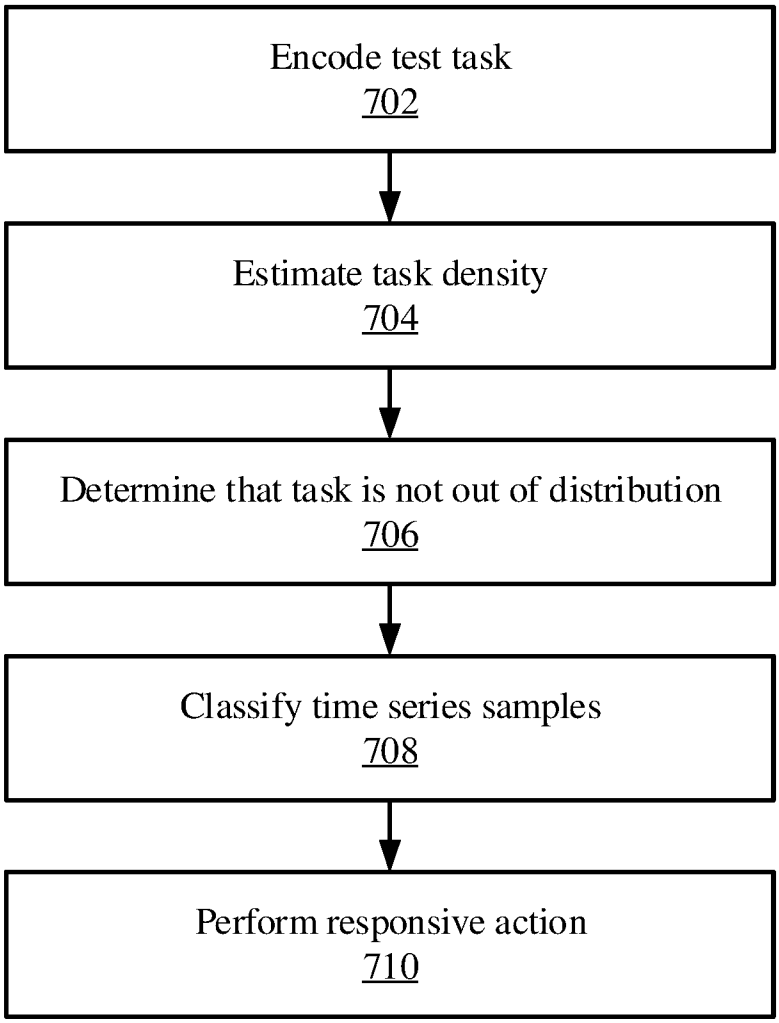
Encode test task
702
Estimate task density
704
Determine that task is not out of distribution
706
Classify time series samples
708
Perform responsive action
710

FIG. 7

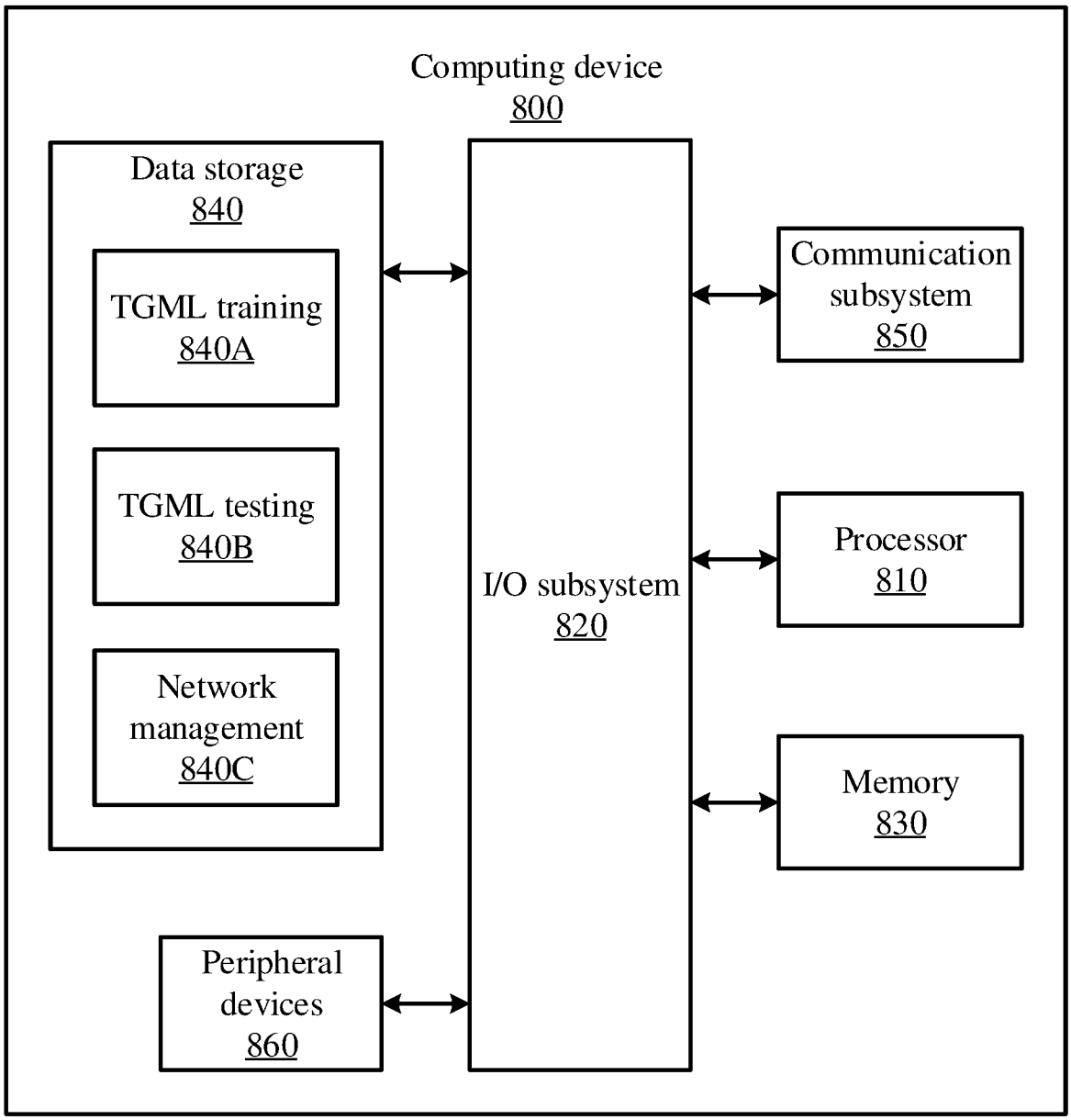
Computing device
800
Data storage
840
TGML training
840A
TGML testing
840B
Network management
840C
I/O subsystem
820
Communication subsystem
850
Processor
810
Memory
830
Peripheral devices
860

FIG. 8

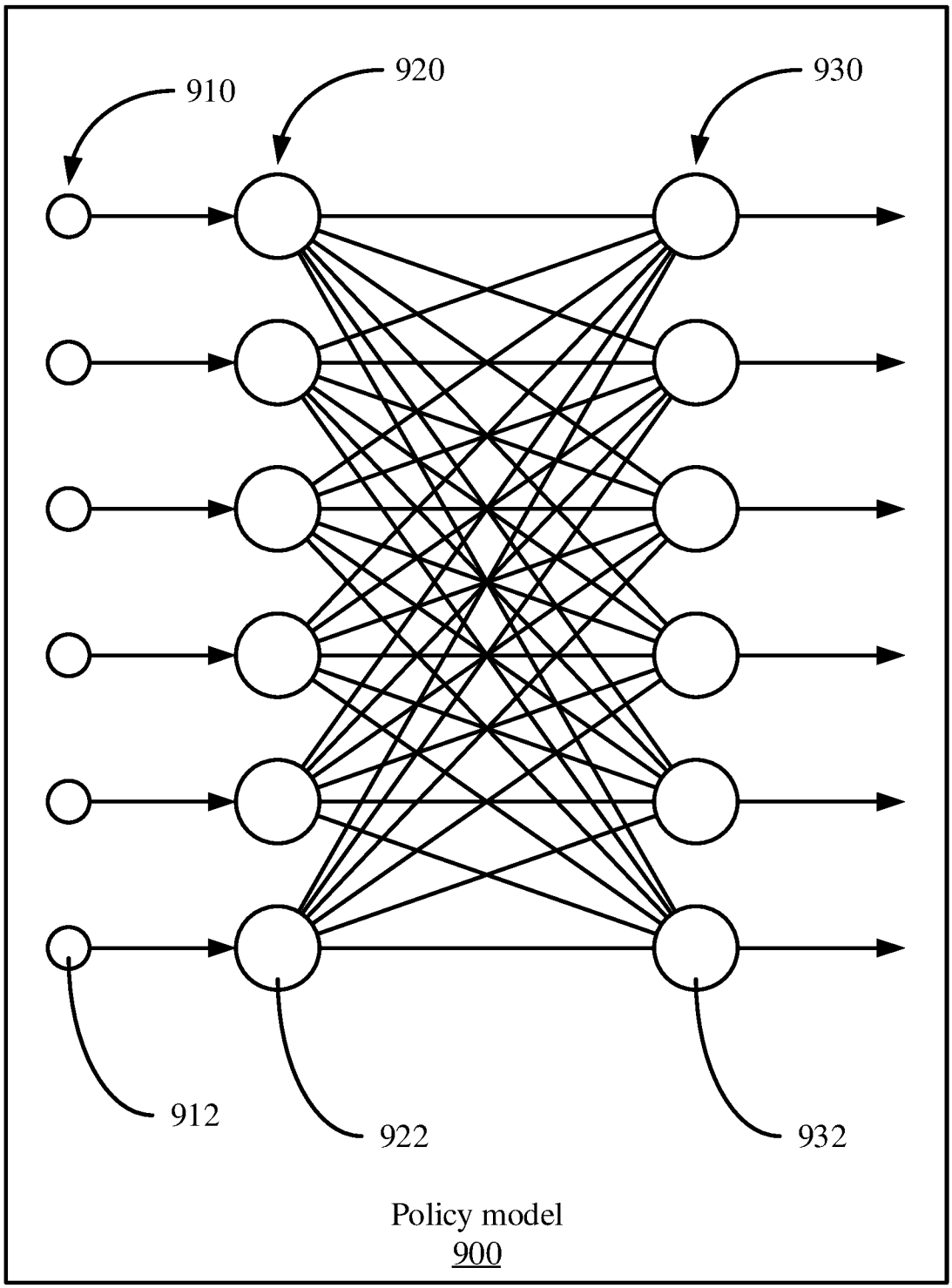

FIG. 9

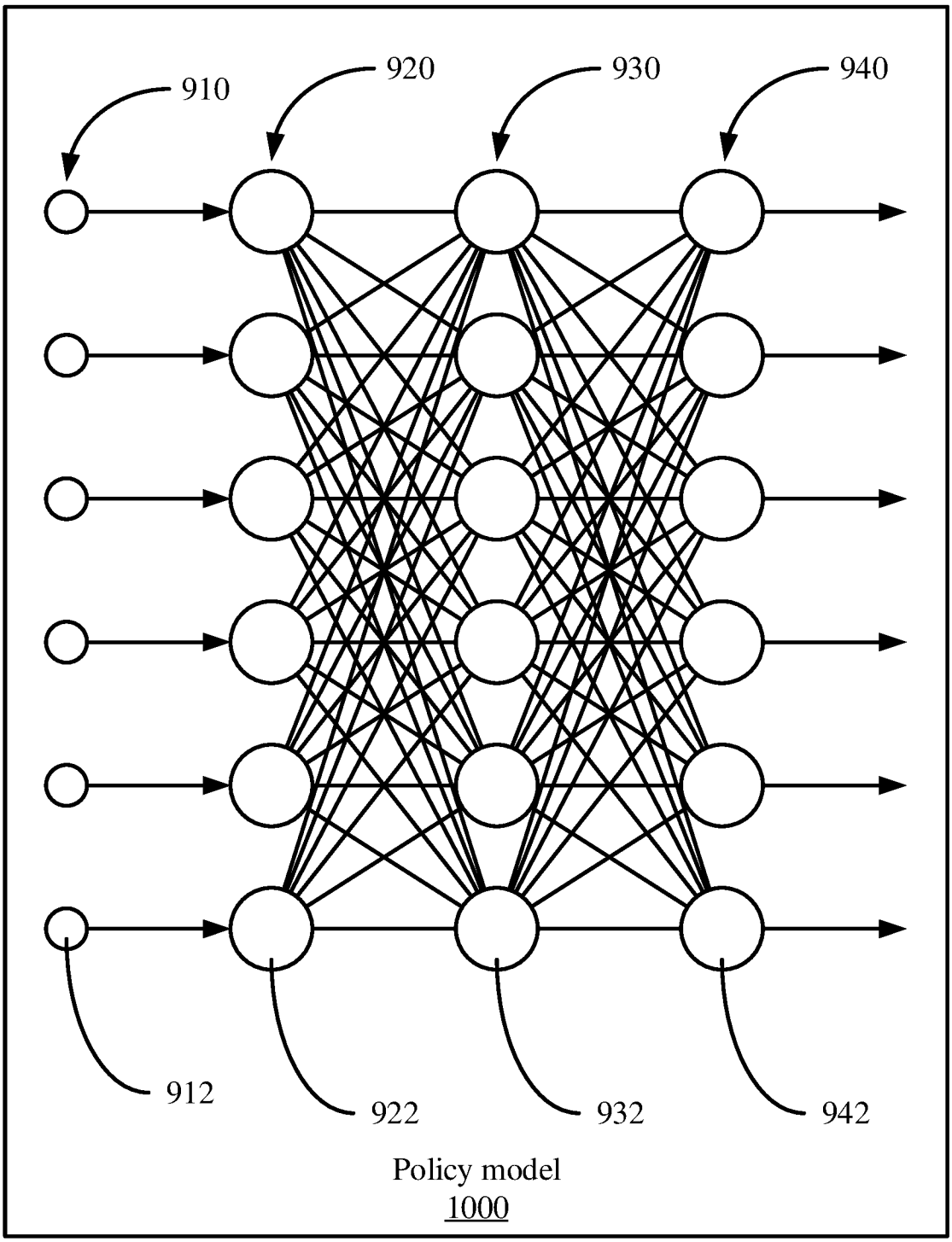

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2023/030773

A. CLASSIFICATION OF SUBJECT MATTER

H04B 10/079(2013.01)i; **G06N 20/00**(2019.01)i; **H04J 14/02**(2006.01)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)

H04B 10/079(2013.01); G05D 1/02(2006.01); G06F 17/16(2006.01); G06K 9/62(2006.01); G06N 20/00(2019.01); G06N 3/04(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: machine learning, training model, SNR, prototype, adjust parameter, task density

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	KR 10-2020-0130153 A (ROBERT BOSCH GMBH) 18 November 2020 (2020-11-18) paragraphs [0008], [0038]	1-5,8-15,18-20
A		6-7,16-17
Y	US 2021-0201116 A1 (DEEPMIND TECHNOLOGIES LIMITED) 01 July 2021 (2021-07-01) paragraph [0042]	1-5,8-15,18-20
A	US 2021-0342637 A1 (TESLA, INC.) 04 November 2021 (2021-11-04) paragraphs [0026]-[0032]; and figure 2	1-20
A	US 2020-0364617 A1 (GOOGLE LLC) 19 November 2020 (2020-11-19) paragraphs [0016]-[0044]; and figure 1	1-20
A	US 2020-0193323 A1 (NEC LABORATORIES EUROPE GMBH) 18 June 2020 (2020-06-18) paragraphs [0053]-[0067]; and figure 1	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
05 December 2023	**05 December 2023**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office **189 Cheongsa-ro, Seo-gu, Daejeon** **35208, Republic of Korea**	**BYUN, Sung Cheal**
Facsimile No. **+82-42-481-8578**	Telephone No. **+82-42-481-8262**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2023/030773

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
KR 10-2020-0130153 A	18 November 2020	CN 111914990 A	10 November 2020
		DE 102019206621 A1	12 November 2020
		EP 3736742 A1	11 November 2020
		JP 2020-184341 A	12 November 2020
		US 11790218 B2	17 October 2023
		US 2020-0356845 A1	12 November 2020
US 2021-0201116 A1	01 July 2021	CN 109328362 A	12 February 2019
		EP 3459017 A1	27 March 2019
		US 10949734 B2	16 March 2021
		US 2017-0337464 A1	23 November 2017
		WO 2017-200597 A1	23 November 2017
US 2021-0342637 A1	04 November 2021	AU 2020-215680 A1	26 August 2021
		CA 3128025 A1	06 August 2020
		CN 113614851 A	05 November 2021
		EP 3918613 A1	08 December 2021
		IL 285153 A	30 September 2021
		IL 285153 D0	30 September 2021
		JP 2022-0523319 A	22 April 2022
		SG 11202108322 A	30 August 2021
		US 10997461 B2	04 May 2021
		US 11748620 B2	05 September 2023
		US 2020-0250473 A1	06 August 2020
		WO 2020-159960 A1	06 August 2020
US 2020-0364617 A1	19 November 2020	CN 111598253 A	28 August 2020
		US 11488067 B2	01 November 2022
		US 2023-0049747 A1	16 February 2023
US 2020-0193323 A1	18 June 2020	None	