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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2012 (29.03.2012)

PCT

(10) International Publication Number
WO 2012/039938 A2

(51) International Patent Classification:
G10L 15/14 (2006.01) *G10L 15/16* (2006.01)

(21) International Application Number:
PCT/US2011/050738

(22) International Filing Date:
7 September 2011 (07.09.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/886,568 21 September 2010 (21.09.2010) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **YU, Dong**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **DENG, Li**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (US). **MOHAMED, Abdel-rahman Samir Abdel-rahman**; c/o Microsoft Corporation, LCA - International Patents, One Microsoft Way, Redmond, Washington 98052-6399 (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, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, 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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: FULL-SEQUENCE TRAINING OF DEEP STRUCTURES FOR SPEECH RECOGNITION

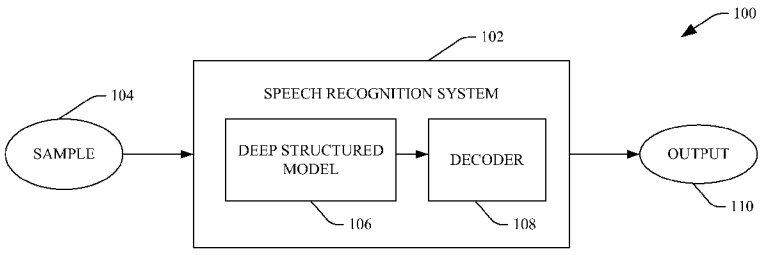

FIG. 1

(57) Abstract: A method is disclosed herein that include an act of causing a processor to access a deep-structured model retained in a computer-readable medium, wherein the deep-structured model comprises a plurality of layers with weights assigned thereto, transition probabilities between states, and language model scores. The method can further include the act of jointly substantially optimizing the weights, the transition probabilities, and the language model scores of the deep-structured model using the optimization criterion based on a sequence rather than a set of unrelated frames.

WO 2012/039938 A2

FULL-SEQUENCE TRAINING OF DEEP STRUCTURES FOR SPEECH RECOGNITION

BACKGROUND

[0001] Speech recognition has been the subject of a significant amount of research and commercial development. For example, speech recognition systems have been incorporated into mobile telephones, desktop computers, automobiles, and the like in order to provide a particular response to speech input provided by a user. For instance, in a mobile telephone equipped with speech recognition technology, a user can speak a name of a contact listed in the mobile telephone and the mobile telephone can initiate a call to the contact. Furthermore, many companies are currently using speech recognition technology to aid customers in connection with identifying employees of a company, identifying problems with a product or service, etc.

[0002] Research in ASR has explored layered architectures to perform speech recognition, motivated partly by the desire to capitalize on some analogous properties in the human speech generation and perception systems. In these studies, learning of model parameters has been one of the most prominent and difficult problems. In parallel with the development in ASR research, recent progresses made in learning methods from neural network research has ignited interest in exploration of deep-structured models. One particular advance is the development of effective learning techniques for deep belief networks (DBNs), which are densely connected, directed belief networks with many hidden layers. In general, DBNs can be considered as a highly complex nonlinear feature extractor with a plurality of layers of hidden units and at least one layer of visible units, where each layer of hidden units learns to represent features that capture higher order correlations in original input data.

[0003] Although DBNs typically have higher modeling power than their more shallow counterparts, learning in DBNs is difficult partly because a back-propagation algorithm often does not perform effectively due to the significantly increased chance of trapping into a local optimum.

[0004] Accordingly, improved learning techniques with respect to DBNs are desirable.

SUMMARY

[0005] The following is a brief summary of subject matter that is described in greater detail herein. This summary is not intended to be limiting as to the scope of the claims.

[0006] Described herein are various technologies pertaining to automatic speech recognition (ASR). With more specificity, various technologies pertaining to utilization of deep-structured models to perform ASR are described herein. With still more specificity, various technologies pertaining to performing full-sequence training of deep-structured models for speech recognition are described herein.

[0007] An exemplary deep-structured model that can be utilized in connection with ASR is a deep belief network (DBN). A pretraining procedure can be undertaken on a DBN, wherein such pretraining procedure can pertain to learning initial weights between layers of variables (visible and hidden) in the DBN. In an example, such pretraining procedure can learn the initial weights for each layer of the DBN greedily by treating each pair of layers in the DBN as a Restricted Boltzmann Machine (RBM).

[0008] Subsequent to the DBN being subjected to pretraining, the DBN weights, transition parameters, and language model (LM) scores can be substantially optimized jointly through utilization of a discriminative training criterion at the sequence level designed for DBNs. More particularly, speech recognition can be referred to as a sequential or full-sequence learning problem, and it has been well known that discriminative information at the sequence level contributes to improving recognition accuracy. In previous approaches, only frame-level information was utilized for training DBN weights, and transition parameters and LM scores are obtained separately.

[0009] Other aspects will be appreciated upon reading and understanding the attached figures and description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Fig. 1 is a functional block diagram of an exemplary system that facilitates performing automatic speech recognition (ASR) through utilization of a Deep Believe Network (DBN).

[0011] Fig. 2 is a functional block diagram of an exemplary system that facilitates initializing weights of a DBN.

[0012] Fig. 3 is a functional block diagram of an exemplary system that facilitates jointly substantially optimizing DBN weights, transition parameters, and language model (LM) scores.

[0013] Fig. 4 is an exemplary DBN.

[0014] Fig. 5 is a flow diagram that illustrates an exemplary methodology for jointly learning DBN weights, transition parameters, and LM scores.

[0015] Fig. 6 is a flow diagram that illustrates an exemplary methodology for jointly learning DBN weights, transition parameters, and LM scores.

[0016] Fig. 7 illustrates an exemplary Deep Hidden Conditional Random Field.

[0017] Fig. 8 is an exemplary computing system.

DETAILED DESCRIPTION

[0018] Various technologies pertaining to automatic speech recognition (ASR) systems will now be described with reference to the drawings, where like reference numerals represent like elements throughout. In addition, several functional block diagrams of example systems are illustrated and described herein for purposes of explanation; however, it is to be understood that functionality that is described as being carried out by certain system components may be performed by multiple components. Similarly, for instance, a component may be configured to perform functionality that is described as being carried out by multiple components, and some steps in methodologies described herein may be omitted, re-ordered, or combined.

[0019] With reference to Fig. 1, an exemplary system 100 that facilitates performing ASR is illustrated. The system 100 includes a speech recognition system 102 that receives a sample 104. The sample can be spoken words from an individual over a particular amount of time (e.g., captured through utilization of a microphone). The sample 104 can be digitized through utilization of an analog to digital converter, and can be subject to some form of normalization if desired. While the examples provided herein indicate that the sample 104 is a spoken utterance, it is to be understood that the system 100 may be configured to perform online handwriting recognition and/or real-time gesture recognition. Thus, the sample 104 may be an online handwriting sample or a video signal describing movement of an object such as a human being.

[0020] The speech recognition system 102 comprises a deep-structured model 106. In an example, the deep-structured model 106 can be a Deep Belief Network (DBN), wherein the DBN is temporally parameter-tied. A DBN is a probabilistic generative model with multiple layers of stochastic hidden units above a single bottom layer of observed variables that represent a data vector. With more specificity, a DBN is a densely connected, directed belief network with many hidden layers for which learning is a difficult problem. The deep-structured model 106 can receive the sample 104 and can output state posterior probabilities with respect to an output unit, which can be a phone, a senone, or some other suitable output unit. As will be described in more detail below, the deep-structured model 106 can be generated through a pretraining procedure, and

thereafter weights of the deep-structured model 106, transition parameters in the deep-structured model 106 and language model scores can be substantially optimized jointly through sequential or full-sequence learning.

[0021] The speech recognition system 102 additionally includes a decoder 108, which can decode output of the deep-structured model to generate an output 110. The output 110, pursuant to an example, can include an indication of a word or word sequence that was received as the sample 104. In another example, the output 110 may be a gesture that pertains to a gesture captured in a video sample. In yet another example, the output 110 can be an indication of a word or word sequence that is being written on a pressure-sensitive screen.

[0022] Pursuant to an example, the speech recognition system 102 can be deployed in a variety of contexts. For instance, the speech recognition system 102 can be deployed in a mobile telephone, such that the mobile telephone can act responsive to spoken commands of a user. In another example, the speech recognition system 102 can be deployed in an automobile, such that the automobile can act responsive to spoken commands of a user. Other systems within which the speech recognition system 102 can be employed include automated transcribing systems, industrial automation systems, banking systems, and other suitable systems that employ ASR technology.

[0023] Now referring to Fig. 2, an exemplary 200 system that facilitates initializing weights of a DBN is illustrated. The system 200 comprises an initializer component 202 that receives a DBN 204. As described previously, DBNs are densely connected, directed belief network with many hidden layers for which learning is a difficult problem. The initializer component 202 can act to learn each layer of the DBN 204 greedily by treating each pair of layers as a Restricted Boltzmann Machine (RBM). The initializer component 202 can access training data in a data repository 206 to perform the aforementioned training. With more detail, an RBM is a particular type of Markov random field (MRF) that has one layer of (typically Bernoulli) stochastic hidden units and one layer of (typically Bernoulli or Gaussian) stochastic visible units. RBMs can be represented as bipartite graphs since all visible units are connected to all hidden units, but there are no visible-visible or hidden-hidden connections.

[0024] In the RBMs, the joint distribution $p(\mathbf{v}, \mathbf{h}; \theta)$ over the visible units $\mathbf{v}$ and hidden units $\mathbf{h}$, given the model parameters θ, can be defined in terms of an energy function $E(\mathbf{v}, \mathbf{h}; \theta)$ of the following algorithm:

$$\mathrm{p}(\mathbf{v},\mathbf{h};\theta) = \frac{exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)}{Z}, \quad (1)$$

where $Z = \sum_u \sum_h exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)$ is a normalization factor or partition function, and the marginal probability that the model assigns to a visible vector **v** can be defined as follows:

$$\mathrm{p}(\mathbf{v};\theta) = \frac{\sum_h exp\left(-\mathrm{E}(\mathbf{v},\mathbf{h};\theta)\right)}{Z} \quad (2)$$

[0025] For a Bernoulli (visible)-Bernoulli (hidden) RBM, the energy is as follows:

$$\mathrm{E}(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j, \quad (3)$$

where w_{ij} represents the symmetric interaction term between visible unit v_i and hidden unit h_j, b_i and a_j represent the bias terms, and V and H are the numbers of visible and hidden units. The conditional probabilities can be calculated as follows:

$$p\left(h_j = 1|\mathbf{v};\theta\right) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right) \quad (4)$$

$$p\left(v_i = 1|\mathbf{h};\theta\right) = \sigma\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i\right), \quad (5)$$

where $\sigma(x) = 1/\left(1 + exp(-x)\right)$.

[0026] Similarly, for a Gaussian-Bernoulli RBM, the energy is as follows after assuming that the variance is unity:

$$\mathrm{E}(\mathbf{v},\mathbf{h};\theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j + \frac{1}{2}\sum_{i=1}^{V}(v_i - b_i)^2 - \sum_{j=1}^{H} a_j h_j, \quad (6)$$

The corresponding conditional probabilities become:

$$p\left(h_j = 1|\mathbf{v};\theta\right) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right) \quad (7)$$

$$p\left(v_i|\mathbf{h};\theta\right) = N\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i, 1\right) \quad (8)$$

where v_i can take real values and can follow a Gaussian distribution with mean $\sum_{j=1}^{H} w_{ij}\, h_j + b_i$ and variance of one. Gaussian-Bernoulli RBMs can be used to convert real-valued stochastic variables to binary stochastic variables which can then be further processed using the Bernoulli-Bernoulli RBMs.

[0027] Following the gradient of the log likelihood $\log p(\boldsymbol{v};\theta)$ the update rule for the weights can be obtained by the initializer component 202 as follows:

$$\Delta w_{ij} = \langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model}, \quad (9)$$

where $\langle v_i h_j\rangle_{data}$ is the expectation observed in the training data and $\langle v_i h_j\rangle_{model}$ is that same expectation under the distribution defined by the DBN 204. Unfortunately, $\langle v_i h_j\rangle_{model}$ can be extremely expensive to compute exactly so the contrastive divergence (CD) approximation to the gradient may be used where $\langle v_i h_j\rangle_{model}$ is replaced by running a Gibbs sampler initialized at the data for one full step.

[0028] From a decoding point of view, the DBN 204 can be treated as a multi-layer perceptron with many layers. The input signal (from the training data) can be processed layer by layer through utilization of equation (4) until the final layer. The final layer can be transformed into a multinomial distribution using the following softmax operation:

$$\mathrm{p}(l = k|\mathbf{h};\theta) = \frac{exp(\sum_{i=1}^{H} \lambda_{ik} h_i + a_k)}{Z(\boldsymbol{h})}, \quad (10)$$

where $l = k$ denotes the input been classified into the k-th class, and λ_{ik} is the weight between hidden unit h_i at the last layer and class label k.

[0029] Pursuant to an example, the initializer component 202 can utilize a conventional frame-level data to train the DBN 204. For instance, the initializer component 202 can train a stack of RBMs in a generative manner, resulting in output of a pretrained DBN 208. As will be described below, the DBN weights, transition parameters, and language model scores can be learned through utilization of a back-propagation algorithm by substantially maximizing the frame-level or utterance-level cross-entropy between the true and the predicted probability distributions over class labels. Furthermore, while the initializer component 202 has been described above as performing pretraining on a DBN in a particular manner, it is to be understood that weights of a DBN can be initialized through other methods, including but not limited to denoising/autoencoding.

[0030] Now referring to Fig. 3, an exemplary system 300 that facilitates jointly substantially optimizing DBN weights, transition parameters, and language model (LM) scores is illustrated. The system 300 includes a receiver component 301 that receives the pretrained DBN 208. A trainer component 302 that is in communication with the receiver component receives the pretrained DBN 208 and the training data in the data store 206 (which can be different training data than what was employed by the initializer component 202 or the same training data employed by the initializer component 202). The trainer component 302 can be configured to jointly substantially optimize weights of the pretrained DBN 208, state transition parameters, and language model scores. For instance, the trainer component 302 can utilize back-propagation to perform such joint fine-tuning of the DBN 208.

[0031] Conventional discriminative back-propagation methods optimize the log posterior probability $p(l_t|v_t)$ of class labels given the current input, both at time-frame t (which may be a fixed local block of frames). This method of training DBNs can be

referred to as a frame-based approach, because it only uses the frame (or frame-block) of an input sample to predict the class labels. The method does not explicitly use the fact that the neighboring frames (or frame-blocks) have smaller distances between the assigned probability distributions over class labels. To take this fact into account, the probability of the whole sequence of labels given the whole utterance $p(l_{1:T}|v_{1:T})$ can be modeled.

[0032] The approach described herein is to consider the top-most layer of the DBN as a linear-chain conditional random field (CRF) with $\boldsymbol{h}_t$ as input features from the lower layer at time t. This model can be viewed as a modification of a deep-structured CRF where the lower multiple layers of CRFs are replaced by DBNs.

[0033] The conditional probability of the full-sequence labels given the full-sequence input features in this sequential model can be given as follows:

$$p(l_{1:T}|v_{1:T}) = p(l_{1:T}|\boldsymbol{h}_{1:T}) = \frac{exp(\sum_{t=1}^{T}\gamma_{ij}\phi_{ij}(l_{t-1},l_t)+\sum_{t=1}^{T}\sum_{d=1}^{D}\lambda_{l_t d}h_{td})}{Z(h_{1:T})} \quad (11)$$

where the transition feature is as follows:

$$\phi_{ij}(l_{t-1},l_t) = \begin{cases} 1 & \text{if } l_{t-1}=i \text{ and } l_t = j \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

γ_{ij} is the parameter associated with this transition feature, h_{td} is the d-th dimension of the hidden unit value at the t-th frame at the last layer $\boldsymbol{h}_t$, and D is the dimension of (or number of units) at that hidden layer.

To optimize the log conditional probability $p(l_{1:T}^n|v_{1:T}^n)$ of the n-th utterance, the trainer component 302 can take the gradient over the activation parameters λ_{kd}, transition parameters γ_{ij}, and M-th-layer weights $w_{ij}^{(M)}$ as follows:

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial \lambda_{kd}} = \sum_{t=1}^{T}\left(\delta(l_t^n = k) - p(l_t^n = k|v_{1:T}^n)\right) h_{td}^{(M),n} \quad (13)$$

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial \gamma_{ij}} = \sum_{t=1}^{T}\left(\delta(l_{t-1}^n = i, l_t^n = j) - p(l_{t-1}^n = i, l_t^n = j|v_{1:T}^n)\right) \quad (14)$$

$$\frac{\partial \log p(l_{1:T}^n|v_{1:T}^n)}{\partial w_{ij}^{(M)}} = \sum_{t=1}^{T}\left(\lambda_{l_t d} - \sum_{k=1}^{K} p(l_t^n = k|v_{1:T}^n)\lambda_{kd}\right) \cdot h_{td}^{(M),n}\left(1 - h_{td}^{(M),n}\right)h_{ti}^{(M-1),n} \quad (15)$$

It can be noted that the gradient $\partial \log p(l_{1:T}^n|v_{1:T}^n)/\partial w_{ij}^{(m)}$ can be considered as back-propagating the error $\delta(l_t^n = k) - p(l_t^n = k|v_{1:T}^n)$ versus $\delta(l_t^n = k) - p(l_t^n = k|v_t^n)$ in a frame-based training algorithm.

[0034] While the basic optimization algorithm with gradient descent can be succinctly described by equations (13), (14), and (15), which compute the gradients in analytical forms, several practical issues can be considered in the algorithm implementation. First, the top-layer CRF's state transition parameters can form a transition matrix, which is different from that of a Hidden Markov Model (HMM). In fact, such state transition parameters are a combination of the transition matrix and bi-phone/senone LM scores. Without proper constraints, the transition matrix may have low likelihoods of being transitioning between states that are prohibited in the left-to-right three-state HMMs even though the training data does not support such transitions. To prevent this from happening so that a sharper model may be built, this constraint can be enforced in the training by setting transition weights that are prohibited in the HMMs to have a very large negative value.

[0035] Second, since the weights in the DBNs are jointly optimized together with CRF's transition parameters, the optimization problem is no longer convex. For this reason, good initialization is crucial. The DBN weights can be initialized by the initializer component 202 (Fig. 2) described above. For example, the transition parameters can be initialized from the combination of the HMM transition matrices and the LM scores, and can be further optimized by tuning the transition features while fixing the DBN weights prior to the trainer component 302 performing joint optimization.

[0036] Third, there are two ways of doing decoding using a DBN that has been trained as described above. A first approach is to feed the log marginal probability $\log p(l_t|v_{1:T})$ as the activation scores to the conventional HMM decoder and use the HMM transition matrices and LM scores in a conventional manner. This approach may work when the full-sequence training can improve the quality of $\log p(l_t|v_{1:T})$. A second approach is to generate the state sequence first and then to map the state sequence to the phoneme/senone sequence. The decoding result may be further improved if insertion penalties are contemplated, which can be integrated into the decoder component 108 (Fig. 1) by modifying the transition parameters by the following:

$$\hat{\gamma}_{ij} = \rho\gamma_{ij} + \varphi \qquad (16)$$

if state i is the final state of a phone and state j is the first state of a phone, where φ is the insertion penalty, and ρ is the scaling factor.

[0037] The pretrained DBN 208 can be configured with the jointly optimized DBN weights, LM scores, and transition probabilities as the parameters. The trainer component 302 can further train the DBN 208 by way of back propagation.

[0038] Now referring to Fig. 4, an exemplary DBN 400 is illustrated. A top level 402 of the DBN can be a linear-chain CRF 404, and the architecture of the DBN 400 can be viewed as shared DBNs unfolding over time (an exemplary shared DBN 406 is illustrated in Fig. 4). The DBN 400 can receive the sample 104 or some derivation thereof, which can be partitioned into a plurality of observed variables 404 over time t. The observed variables 408 can represent data vectors at different instances in time. The DBN 400 further comprises multiple layers of stochastic hidden units 410. The DBN 400 has undirected connections 412 between the top two layers of the stochastic hidden units 410 and directed connections 414 to all other layers from the layers above. Weights w can be initially assigned to the directed and undirected connections 412 and 414, respectively, during the pretraining described above. λ_{ik} is the weight between hidden unit h_i at the last layer in the DBN 400 and class label k, and γ_{ij} are transition probabilities between classes. In this exemplary embodiment, the DBN 400 can be trained such that the output units in the uppermost layer (the Mth layer) can be modeled as a phonetic unit or subunit, such as a phone or senone.

[0039] With reference now to Figs. 5 and 6, exemplary methodologies are illustrated and described. While the methodologies are described as being a series of acts that are performed in a sequence, it is to be understood that the methodology is not limited by the order of the sequence. For instance, some acts may occur in a different order than what is described herein. In addition, an act may occur concurrently with another act. Furthermore, in some instances, not all acts may be required to implement a methodology described herein.

[0040] Moreover, the acts described herein may be computer-executable instructions that can be implemented by one or more processors and/or stored on a computer-readable medium or media. The computer-executable instructions may include a routine, a sub-routine, programs, a thread of execution, and/or the like. Still further, results of acts of the methodologies may be stored in a computer-readable medium, displayed on a display device, and/or the like. The computer-readable medium may be a non-transitory medium, such as memory, hard drive, CD, DVD, flash drive, or the like.

[0041] With reference solely to Fig. 5, an exemplary methodology 500 that facilitates training a deep-structured model for utilization in a speech recognition system is illustrated. The methodology 500 begins at 502, and at 504 parameters of a deep-structured model are provided through a pretraining step. For example, weights between layers of a DBN can be initialized during such pretraining step. At 506, labeled training data is provided to the deep structure, wherein the labeled training data may be labeled words or word sequences, labeled gestures, labeled handwriting samples, etc. At 508, weights between layers in the deep-structured model, language model parameters, and state transition probabilities are jointly substantially optimized, such that the resultant trained deep-structured model can be commissioned into a speech recognition system. The methodology 500 completes at 510.

[0042] Turning now to Fig. 6, an exemplary methodology 600 that facilitates training a DBN for utilization in an automated speech recognition system is illustrated. The methodology 600 starts at 602, and at 604 each layer of a DBN that is configured for utilization in an automated speech recognition system is greedily learned. At 606, the log conditional probabilities of the output states/sequences of the DBN are substantially optimized through utilization of training data. At 608, weights in the DBN, transition parameters in the DBN, and language model scores are substantially simultaneously optimized based at least in part upon the log of the conditional probabilities of output states/sequences produced by the DBN. The methodology 600 completes at 610.

[0043] The systems and methodologies shown and described above have generally referred to utilizing a DBN in a speech recognition system; as indicated above, however, other deep structures can be employed. An exemplary deep structure that can be utilized is a Deep Hidden Conditional Random Field (DHCRF). Referring to Fig. 7, an exemplary DHCRF 700 is illustrated. In an example, the *N*th layer of the DHCRF can be a Hidden Conditional Random Field (HCRF), and the intermediate layers can be zero-th order CRFs that do not use state transition features.

[0044] In the exemplary DHCRF 700, the observation sequence $\boldsymbol{o}^{\boldsymbol{j}}$ at layer j consists of two parts: the preceding layer's observation sequence $\boldsymbol{o}^{\boldsymbol{j-1}}$ and the frame-level log marginal posterior probabilities $\log p\left(s_t^{j-1} \middle| \boldsymbol{o}^{\boldsymbol{j-1}}\right)$ computed from the preceding layer $j-1$, where s_t^{j-1} is the state value at layer $j-1$. The raw observations at the first layer can be denoted as $\boldsymbol{o} = [\boldsymbol{o_t}], t = 1, \cdots, T$.

[0045] Both parameter estimation and sequence inference in the DHCRF can be carried out bottom-up, layer by layer. The final layer's state sequence conditional probability can be shown as follows:

$$p(w|\boldsymbol{o}^N;\boldsymbol{\lambda}^N) = \frac{1}{z(\boldsymbol{o}^N;\boldsymbol{\lambda}^N)}\sum_{s^N\in w} exp\left((\boldsymbol{\lambda}^N)^T\boldsymbol{f}(w,s^N,\boldsymbol{o}^N)\right) \quad (17)$$

where N is the total number of layers, $(\cdot)^T$ is the transposition of $(\cdot)$, $\boldsymbol{o}^N = (\boldsymbol{o}_1^N,\cdots,\boldsymbol{o}_T^N)$ is the observation sequence at the final layer, w is the output sequence (senone, phoneme, word, etc.), $s^N = (s_1^N,\cdots,s_T^N)$ is a hypothesized state sequence, $\boldsymbol{f}(w,s^N,\boldsymbol{o}^N) = [f_1(w,s^N,\boldsymbol{o}^N),\cdots,f_T(w,s^N,\boldsymbol{o}^N)]^T$ is the feature vector at the final layer, $\boldsymbol{\lambda}^N = [\lambda_1^N,\cdots,\lambda_T^N]^T$ is the model parameter (weight vector), and $z(\boldsymbol{o}^N;\boldsymbol{\lambda}^N) = \sum_{w,s^N\in w} exp\left((\boldsymbol{\lambda}^N)^T\boldsymbol{f}(w,s^N,\boldsymbol{o}^N)\right)$ is the partition function (normalization factor) to ensure probabilities $p(w|\boldsymbol{o}^N;\boldsymbol{\lambda}^N)$ sum to one. It can be ascertained that invalid sequences can be ruled out by summing over valid phoneme or word sequences only.

[0046] In contrast to the final layer, the state conditional probabilities at the intermediate layer j can be as follows:

$$p(s^j|\boldsymbol{o}^j;\boldsymbol{\lambda}^j) = \frac{1}{z(\boldsymbol{o}^j;\boldsymbol{\lambda}^j)} exp\left((\boldsymbol{\lambda}^j)^T\boldsymbol{f}(s^j,\boldsymbol{o}^j)\right). \quad (18)$$

This is different from (17) in two ways. First, transition features are not used in (18) and observation features $\boldsymbol{f}(s^j,\boldsymbol{o}^j)$ can be simplified to $[\boldsymbol{f}(s_t^j,\boldsymbol{o}_t^j)]_{t=1,\cdots,T}$ which is defined below. Second, there is no summation over state sequences with all possible segmentations in (18).

[0047] Weights of the DHCRF can be learned utilizing a combination of supervised and unsupervised learning. The training supervision of the DHCRF 700 is available only at the final layer and can be directly determined by the problem to be solved. For example, in the phonetic recognition task, the phoneme sequence w is known at the final layer during the training phase. Parameter estimation at the final layer can thus be carried out in a supervised manner. The supervision, however, is not available for the intermediate layers, which play the role of converting original observations to some intermediate abstract representations. For this reason, an unsupervised approach can be utilized to learn parameters in the intermediate layers.

[0048] There are several approaches to learning the intermediate layer representations in the DHCRF 700. For example, the intermediate layer learning problem can be cast into a multi-objective programming (MOP) problem in which the average frame-level conditional entropy is minimized and the state occupation entropy is

maximized at a substantially similar time. Minimizing the average frame-level conditional entropy can force the intermediate layers to be sharp indicators of subclasses (or clusters) for each input vector, while maximizing the occupation entropy guarantees that the input vectors be represented distinctly by different intermediate states. The MOP optimization algorithm alternates the steps in optimizing these two contradictory criteria until no further improvement in the criteria is possible or the maximum number of iterations is reached. The MOP optimization, however, can become difficult when the number of classes in the intermediate layers becomes higher (as in a phone recognition task) since it is hard to control when to switch to optimize the other criterion given the vastly increased probability of being trapped into a local optimum.

[0049] Alternatively, a GMM-based algorithm can be employed to learn parameters in the intermediate layers of the DHCRF 700. This algorithm can utilize a layer-by-layer approach: once a lower layer is trained, the parameters of that layer are fixed and the observation sequences of the next layer are generated using the newly trained lower-layer parameters. This process can continue until all the layers are trained.

[0050] With more specificity, to learn the parameters of an intermediate layer, a single GMM with diagonal covariance (initialized from the corresponding HMM model which is optimized using the Gaussian splitting strategy) can be trained. The following can then be assigned as the state value to each observation frame $\boldsymbol{o}_t^j$ at layer j by assuming each Gaussian component is a state, where $\boldsymbol{\mu}_i^j$ and $\boldsymbol{\Sigma}_i^j$ are the mean and variance of the i-th Gaussian component at layer j:

$$s_t^j = \text{argmax}_i \, N\left(\boldsymbol{o}_t^j; \boldsymbol{\mu}_i^j, \boldsymbol{\Sigma}_i^j\right) \tag{19}$$

The parameters of the CRF at layer j can then be learned by maximizing the regularized log-conditional probability as follows:

$$J_1\left(\boldsymbol{\lambda}^j\right) = \sum_k \sum_t \log p\left(\mathrm{s}_\mathrm{t}^{(k),j} | \mathbf{o}_\mathrm{t}^{(k),j}; \boldsymbol{\lambda}^j\right) - \frac{\|\boldsymbol{\lambda}^j\|_1}{\sigma_1} - \frac{\|\boldsymbol{\lambda}^j\|_2^2}{\sigma_2} \tag{20}$$

where k is the utterance ID, $\|.\|_1$ is a L1-norm to enforce sparseness of the parameters associated with each state value, $\|.\|_2^2$ is the square of L2-norm to give preference to smaller weights, and σ_1 and σ_2 are positive values to determine the importance of each regularization term. A regularized dual averaging method can be used to solve this optimization problem with L1/L2 regularization terms.

[0051] Pursuant to an example, transition features may not be used in the intermediate layers. Instead, only the first- and second-order observation features can be used as follows:

$$\boldsymbol{f}_{s\prime}^{(M1)}(s_t, \boldsymbol{o}_t) = \delta(s_t = s')\boldsymbol{o}_t \qquad \forall s' \qquad (21)$$

$$\boldsymbol{f}_{s\prime}^{(M2)}(s_t, \boldsymbol{o}_t) = \delta(s_t = s')\boldsymbol{o}_t \circ \boldsymbol{o}_t \qquad \forall s' \qquad (22)$$

where $\circ$ is an element-wise product.

[0052] The final layer of the DHCRF 700 can be trained to optimize the following in a supervised manner:

$$J_2(\boldsymbol{\lambda}^N) = \sum_k \log p\left(w^{(k)} | \boldsymbol{o}^{(k),N}\right) - \frac{\|\lambda^N\|_1}{\sigma_1} - \frac{\|\lambda^N\|_2^2}{\sigma_2} \qquad (23)$$

where $w^{(k)}$ is the label for the output unit for the k-th utterance without segmentation information. In the final layer, the following can be used as features:

$$\boldsymbol{f}_{w^{"}w'}^{(LM)}(w, s, \boldsymbol{o}) = [\delta(w_{i-1} = w^{"})\delta(w_i = w')]_{\mathrm{i}=1,\cdots,\mathrm{I}} \qquad \forall w^{"}w' \qquad (24)$$

$$\boldsymbol{f}_{s^{"}s\prime}^{(Tr)}(w, s, \boldsymbol{o}) = [\delta(s_{t-1} = s^{"})\delta(s_t = s')]_{\mathrm{t}=1,\cdots,\mathrm{T}} \qquad \forall s^{"}, s' \qquad (25)$$

$$\boldsymbol{f}_{s\prime}^{(M1)}(w, s, \boldsymbol{o}) = [\delta(s_t = s')\boldsymbol{o}_t]_{\mathrm{t}=1,\cdots,\mathrm{T}} \qquad \forall s' \qquad (26)$$

$$\boldsymbol{f}_{s\prime}^{(M2)}(w, s, \boldsymbol{o}) = [\delta(s_t = s')\boldsymbol{o}_t \circ \boldsymbol{o}_t]_{\mathrm{t}=1,\cdots,\mathrm{T}} \qquad \forall s' \qquad (27)$$

where $\delta(x) = 1$ if x is true, and $\delta(x) = 0$ otherwise. $\boldsymbol{f}_{w^{"}w'}^{(LM)}(w, s, \boldsymbol{o})$ are bi-gram language model (LM) features in which each output unit sequence w is consisted of I output units (e.g., senones, phonemes, or words), $\boldsymbol{f}_{s^{"}s\prime}^{(Tr)}(w, s, \boldsymbol{o})$ are state transition features, and $\boldsymbol{f}_{s\prime}^{(M1)}(w, s, \boldsymbol{o})$ and $f_{s\prime}^{(M2)}(w, s, \boldsymbol{o})$ are the first- and second-order statistics generated from the observations, respectively.

[0053] Now referring to Fig. 8, a high-level illustration of an example computing device 800 that can be used in accordance with the systems and methodologies disclosed herein is illustrated. For instance, the computing device 800 may be used in a system that supports ASR. In another example, at least a portion of the computing device 800 may be used in a system that supports training a DBN. The computing device 800 includes at least one processor 802 that executes instructions that are stored in a memory 804. The memory 804 may be or include RAM, ROM, EEPROM, Flash memory, or other suitable memory. The instructions may be, for instance, instructions for implementing functionality described as being carried out by one or more components discussed above or instructions for implementing one or more of the methods described above. The processor 802 may access the memory 804 by way of a system bus 806. In addition to

storing executable instructions, the memory 804 may also store a training data set, a validation data set, a DBN, etc.

[0054] The computing device 800 additionally includes a data store 808 that is accessible by the processor 802 by way of the system bus 806. The data store may be or include any suitable computer-readable storage, including a hard disk, memory, etc. The data store 808 may include executable instructions, a DBN, a training data set, a validation data set, etc. The computing device 800 also includes an input interface 810 that allows external devices to communicate with the computing device 800. For instance, the input interface 810 may be used to receive instructions from an external computer device, from a user, etc. The computing device 800 also includes an output interface 812 that interfaces the computing device 800 with one or more external devices. For example, the computing device 800 may display text, images, etc. by way of the output interface 812.

[0055] Additionally, while illustrated as a single system, it is to be understood that the computing device 800 may be a distributed system. Thus, for instance, several devices may be in communication by way of a network connection and may collectively perform tasks described as being performed by the computing device 800.

[0056] As used herein, the terms "component" and "system" are intended to encompass hardware, software, or a combination of hardware and software. Thus, for example, a system or component may be a process, a process executing on a processor, or a processor. Additionally, a component or system may be localized on a single device or distributed across several devices. Furthermore, a component or system may refer to a portion of memory and/or a series of transistors.

[0057] It is noted that several examples have been provided for purposes of explanation. These examples are not to be construed as limiting the hereto-appended claims. Additionally, it may be recognized that the examples provided herein may be permutated while still falling under the scope of the claims.

CLAIMS

What is claimed is:

1. A method comprising the following computer-executable acts:

causing a processor to access a deep-structured model retained in a computer-readable medium, wherein the deep-structured model comprises a plurality of layers with weights assigned thereto, transition probabilities between states, and language model scores; and

jointly optimizing the weights, the transition probabilities, and the language model scores of the deep-structured model.

2. The method of claim 1, wherein the deep-structured model is a deep belief network (DBN).

3. The method of claim 2, wherein the DBN is configured to perform one of automatic speech recognition, automatic gesture recognition, automatic human body action recognition, or automatic online handwriting recognition.

4. The method of claim 2, wherein the DBN is a probabilistic generative model that comprises multiple layers of stochastic hidden units above a single bottom layer of observed variables that represent a data vector.

5. The method of claim 1, wherein the deep-structured model comprises a plurality of hidden stochastic layers, and further comprising pretraining the deep-structured model, wherein pretraining comprises utilizing an unsupervised algorithm to initialize weights of connections between the hidden stochastic layers.

6. The method of claim 5, further comprising utilizing back-propagation to jointly substantially optimize the weights, the transition probabilities, and the language model scores of the deep-structured model.

7. The method of claim 5, wherein pretraining comprises treating pairs of layers in the deep-structured model as a Restricted Boltzmann Machine.

8. The method of claim 1, wherein the deep-structured model is a deep hidden conditional random field (DHCRF).

9. A computer-implemented system comprising:

a processor; and

a memory that comprises a plurality of components that are executable by the processor, the components comprising:

a receiver component that receives a pretrained deep-structured model, wherein the deep-structured model comprises a plurality of layers, weights between the layers, transition parameters, and language model scores; and

a trainer component that jointly substantially optimize weights of the pretrained deep-structured model, state transition parameters of the pretrained deep-structured model, and language model scores of the pretrained deep-structured model.

10. The system of claim 9, wherein the pretrained deep-structured model is trained for speech recognition.

11. The system of claim 9, wherein the pretrained deep-structured model is a deep belief network (DBN).

12. The system of claim 11, wherein the DBN is a probabilistic generative model that comprises multiple layers of stochastic hidden units above a single bottom layer of observed variables that represent a data vector.

13. The system of claim 11, wherein the top-most layer of the DBN is a linear-chain conditional random field (CRF).

14. The system of claim 9, wherein the components further comprise an initializer component that initializes weights of a deep-structured model to generate the pretrained deep-structured model.

15. The system of claim 9, wherein the trainer component determines a conditional probability of a full-sequence of labels of the deep-structured model in connection with substantially optimizing the weights, transition parameters, and language model scores to generate the trained deep-structured model.

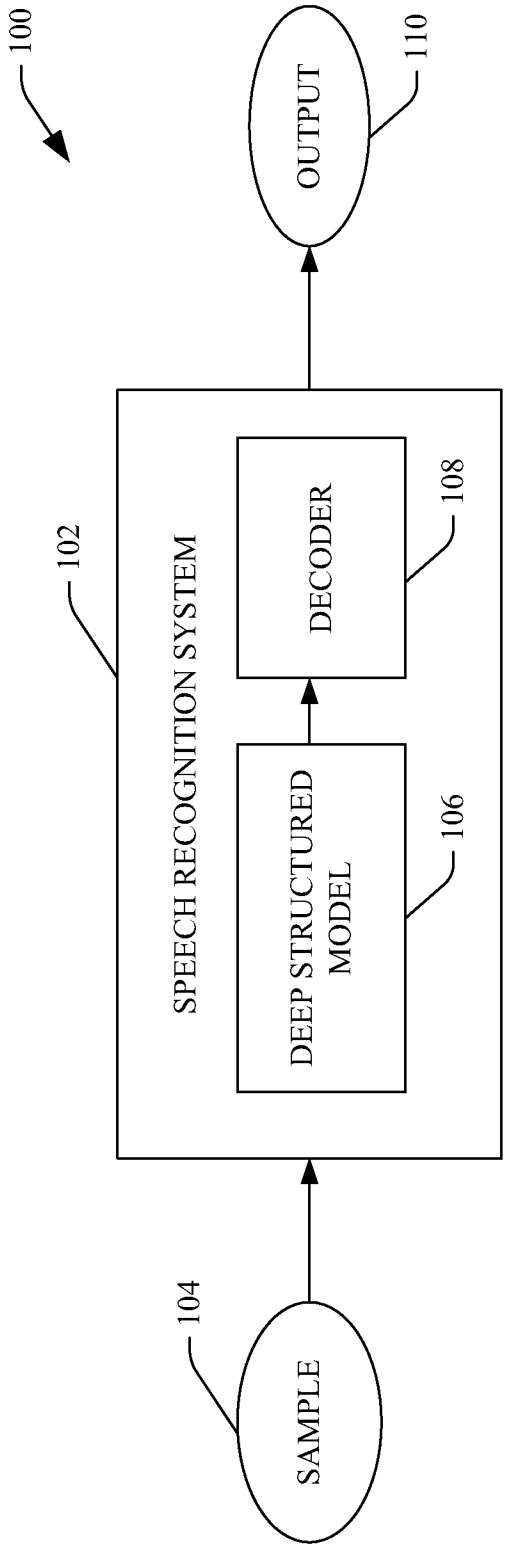
100
OUTPUT
110
102
SPEECH RECOGNITION SYSTEM
DECODER
108
DEEP STRUCTURED MODEL
106
SAMPLE
104

FIG. 1

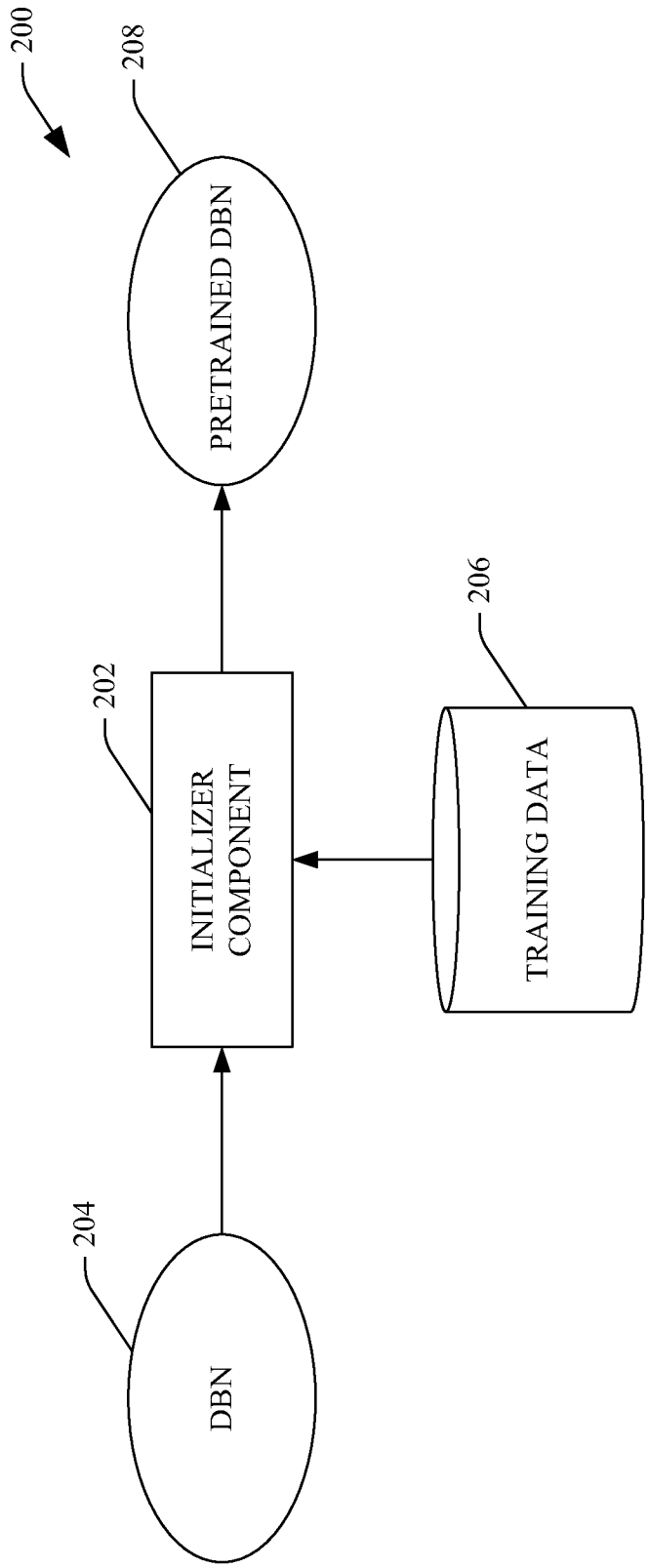
200
208
PRETRAINED DBN
206
202
INITIALIZER COMPONENT
TRAINING DATA
204
DBN

FIG. 2

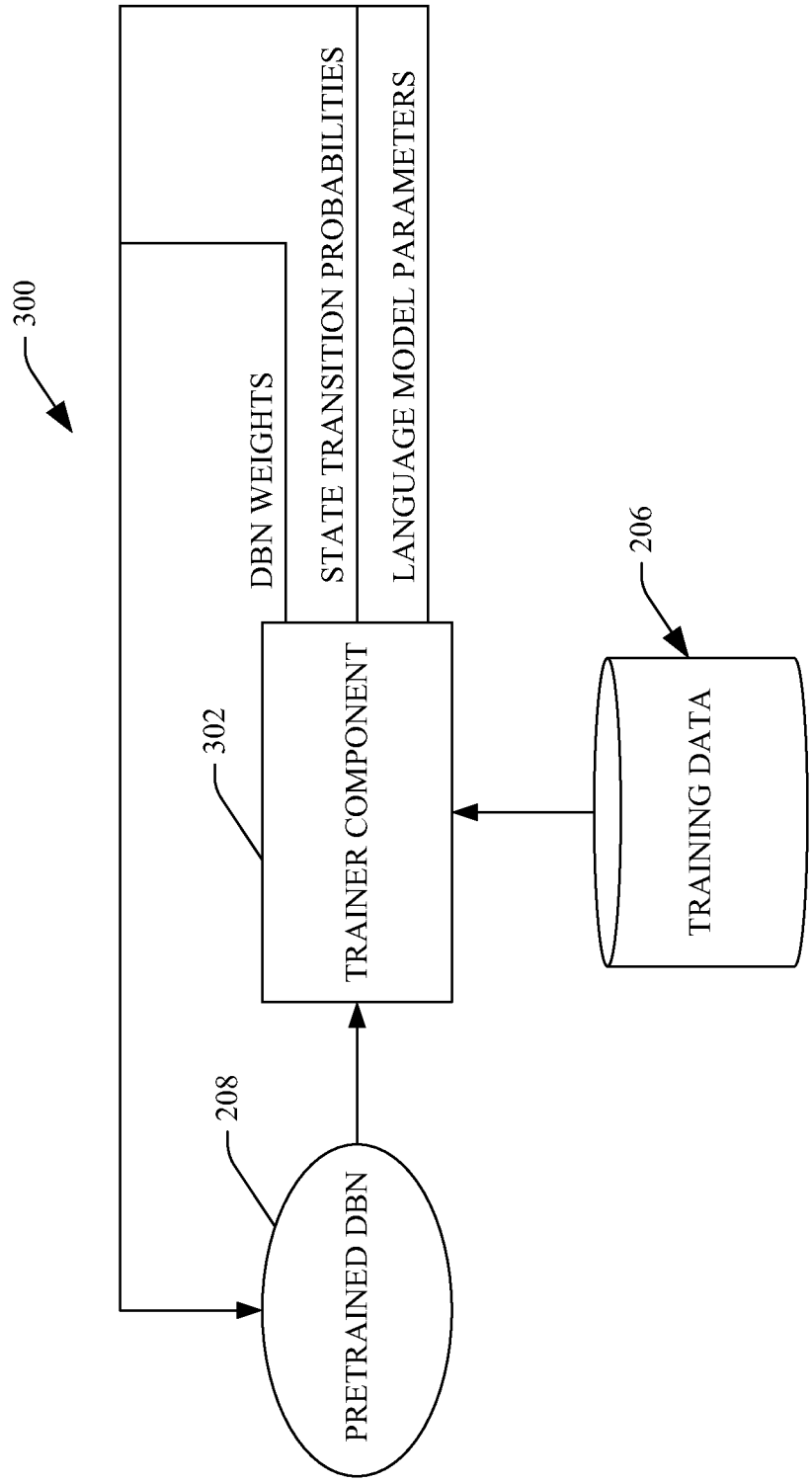
300
DBN WEIGHTS
STATE TRANSITION PROBABILITIES
LANGUAGE MODEL PARAMETERS
302
TRAINER COMPONENT
206
TRAINING DATA
208
PRETRAINED DBN

FIG. 3

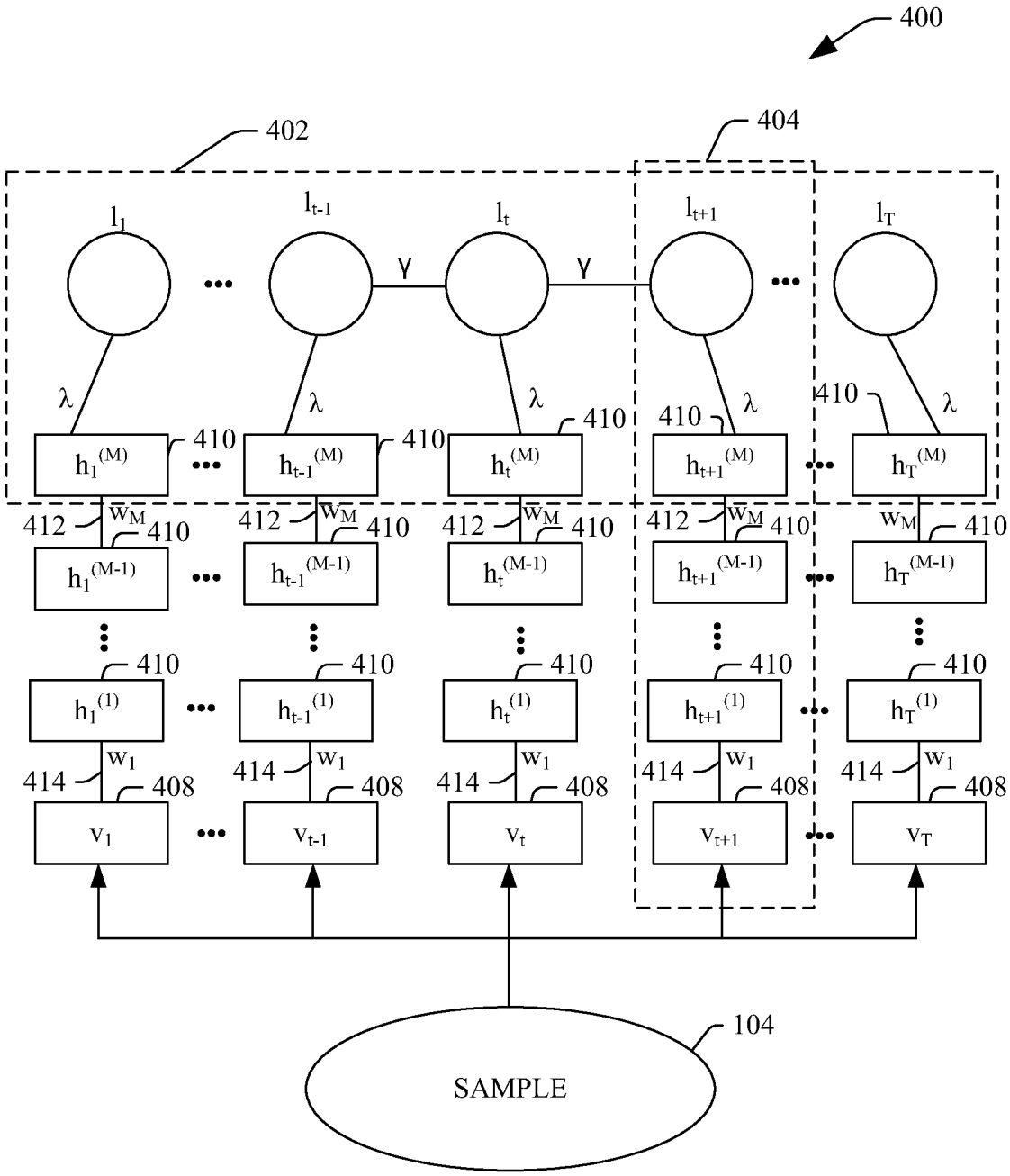
400
402
404
l_1
l_{t-1}
l_t
l_{t+1}
l_T
γ
γ
λ
λ
λ
λ
λ
410
$h_1^{(M)}$
$h_{t-1}^{(M)}$
$h_t^{(M)}$
$h_{t+1}^{(M)}$
$h_T^{(M)}$
412
w_M
$h_1^{(M-1)}$
$h_{t-1}^{(M-1)}$
$h_t^{(M-1)}$
$h_{t+1}^{(M-1)}$
$h_T^{(M-1)}$
$h_1^{(1)}$
$h_{t-1}^{(1)}$
$h_t^{(1)}$
$h_{t+1}^{(1)}$
$h_T^{(1)}$
414
w_1
408
v_1
v_{t-1}
v_t
v_{t+1}
v_T
104
SAMPLE

FIG. 4

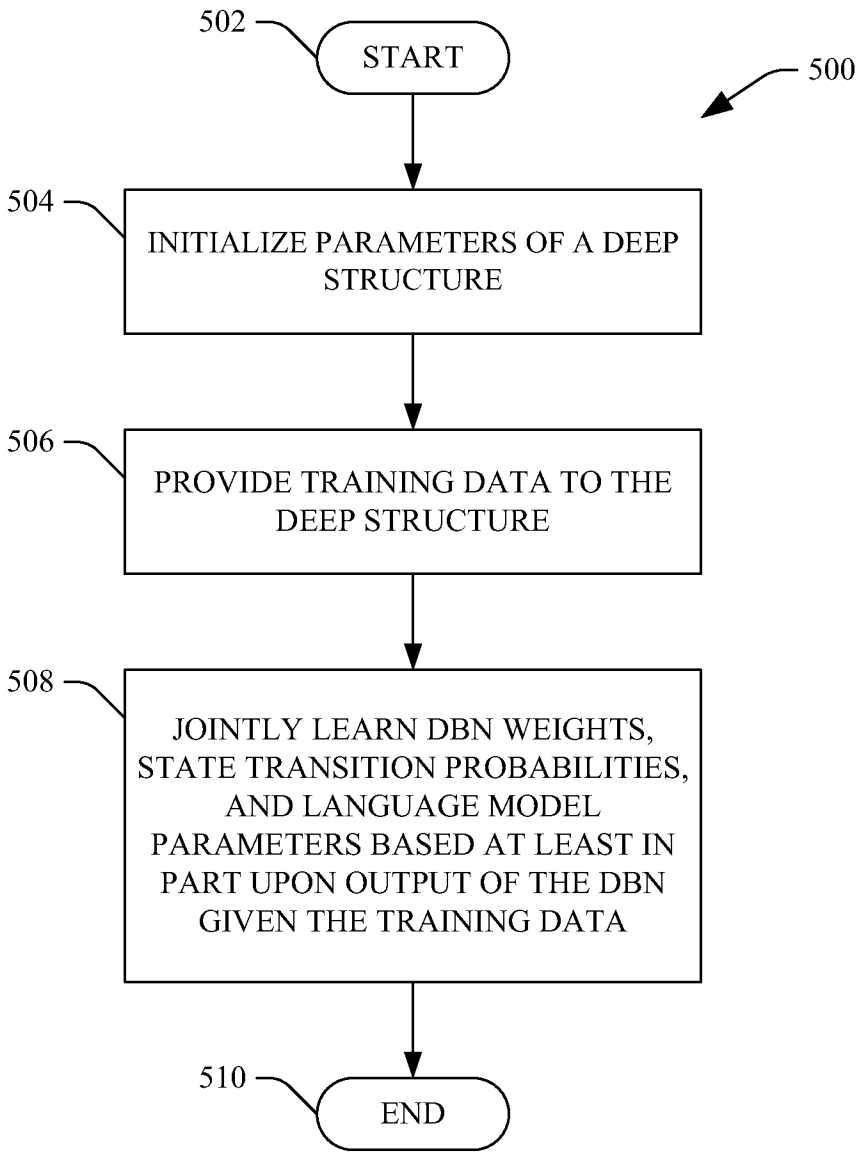
502
START
500
504
INITIALIZE PARAMETERS OF A DEEP STRUCTURE
506
PROVIDE TRAINING DATA TO THE DEEP STRUCTURE
508
JOINTLY LEARN DBN WEIGHTS, STATE TRANSITION PROBABILITIES, AND LANGUAGE MODEL PARAMETERS BASED AT LEAST IN PART UPON OUTPUT OF THE DBN GIVEN THE TRAINING DATA
510
END

FIG. 5

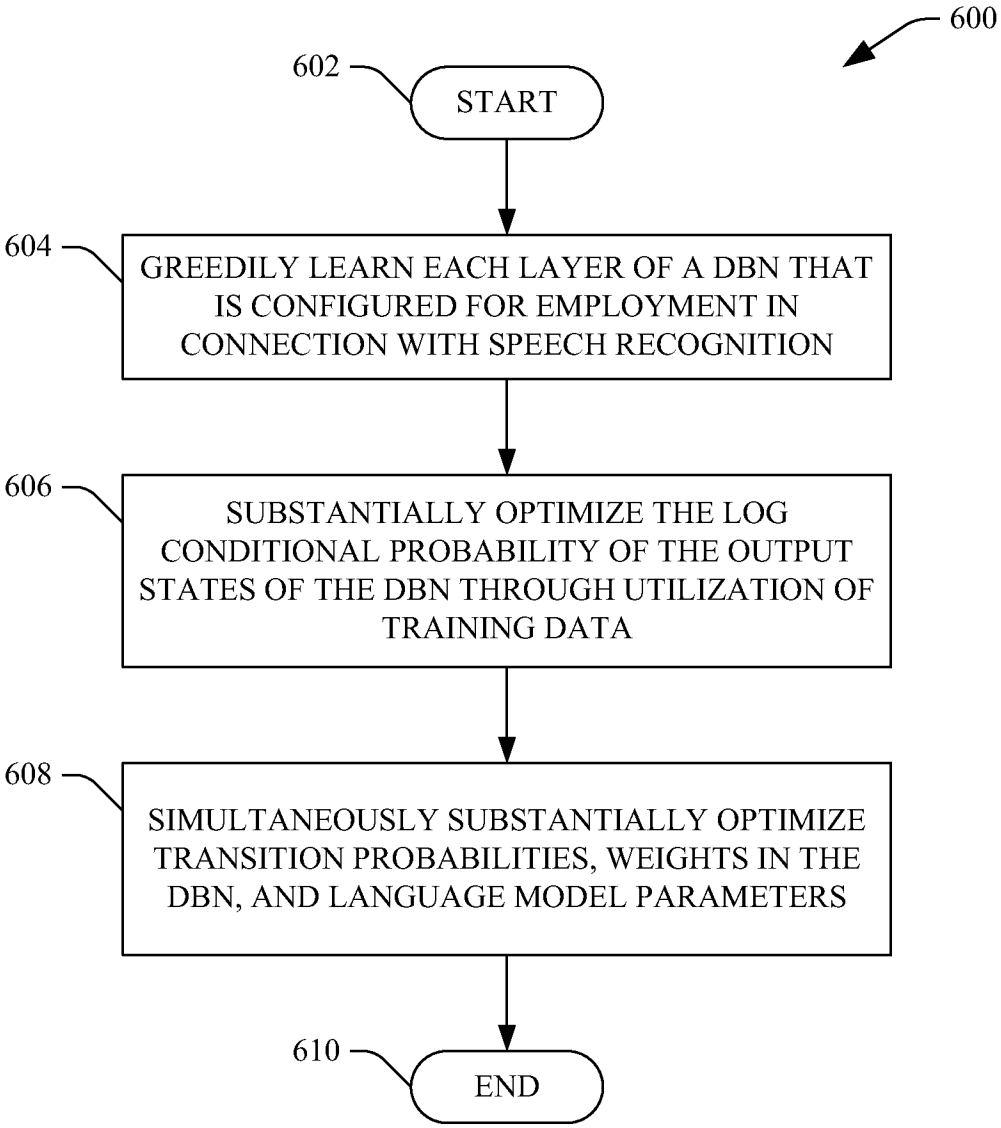
600
602
START
604
GREEDILY LEARN EACH LAYER OF A DBN THAT IS CONFIGURED FOR EMPLOYMENT IN CONNECTION WITH SPEECH RECOGNITION
606
SUBSTANTIALLY OPTIMIZE THE LOG CONDITIONAL PROBABILITY OF THE OUTPUT STATES OF THE DBN THROUGH UTILIZATION OF TRAINING DATA
608
SIMULTANEOUSLY SUBSTANTIALLY OPTIMIZE TRANSITION PROBABILITIES, WEIGHTS IN THE DBN, AND LANGUAGE MODEL PARAMETERS
610
END

FIG. 6

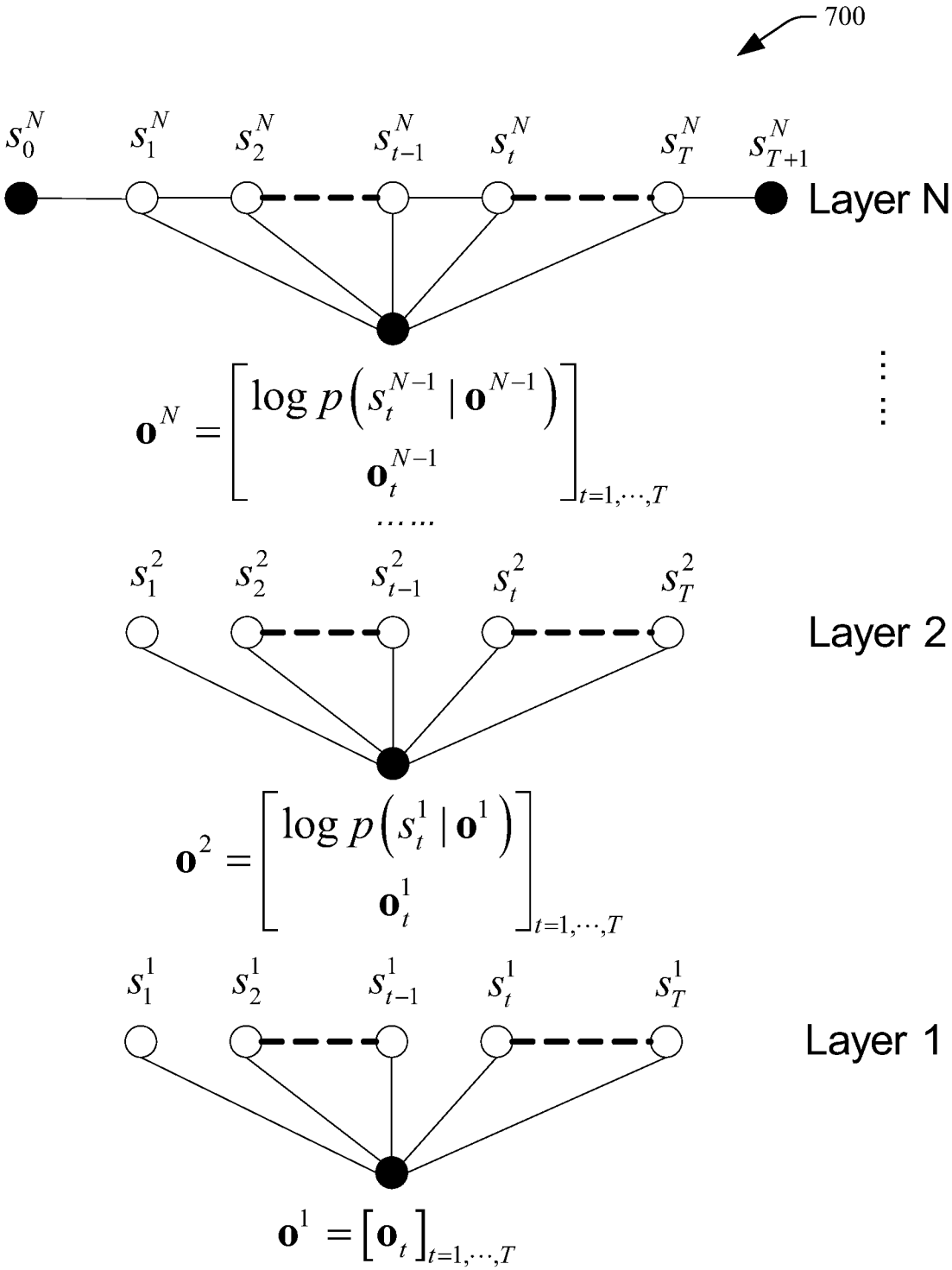
700
s_0^N s_1^N s_2^N s_{t-1}^N s_t^N s_T^N s_{T+1}^N
Layer N
$\mathbf{o}^N = \left[\begin{array}{c} \log p\left(s_t^{N-1} \mid \mathbf{o}^{N-1}\right) \\ \mathbf{o}_t^{N-1} \end{array}\right]_{t=1,\cdots,T}$
s_1^2 s_2^2 s_{t-1}^2 s_t^2 s_T^2
Layer 2
$\mathbf{o}^2 = \left[\begin{array}{c} \log p\left(s_t^1 \mid \mathbf{o}^1\right) \\ \mathbf{o}_t^1 \end{array}\right]_{t=1,\cdots,T}$
s_1^1 s_2^1 s_{t-1}^1 s_t^1 s_T^1
Layer 1
$\mathbf{o}^1 = \left[\mathbf{o}_t\right]_{t=1,\cdots,T}$

FIG. 7

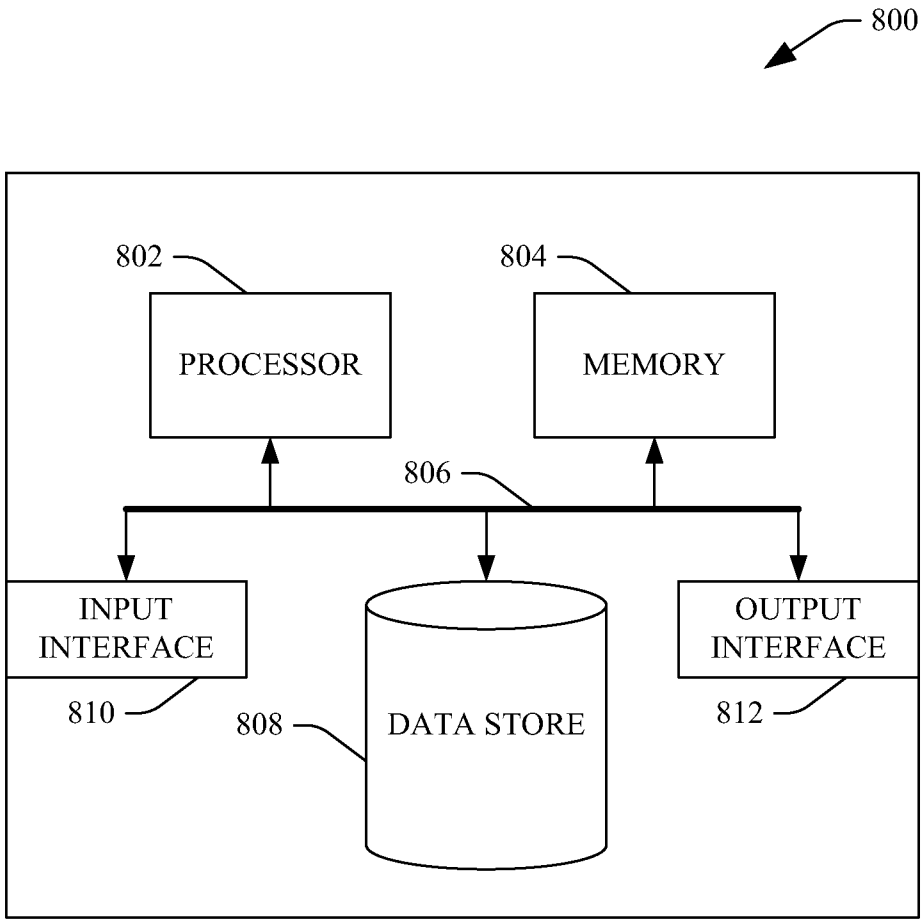
800
802
PROCESSOR
804
MEMORY
806
INPUT INTERFACE
810
808
DATA STORE
OUTPUT INTERFACE
812

FIG. 8