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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 December 2014 (24.12.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/205231 A1

(51) International Patent Classification:
G06T 7/20 (2006.01) *G06K 9/00* (2006.01)

(21) International Application Number:
PCT/US2014/043206

(22) International Filing Date:
19 June 2014 (19.06.2014)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/836,845 19 June 2013 (19.06.2013) US

(71) Applicant: **THE REGENTS OF THE UNIVERSITY OF MICHIGAN** [US/US]; Office of Technology Transfer, 1600 Huron Parkway, 2nd Floor, Ann Arbor, Michigan 48109-2590 (US).

(72) Inventors: **LEE, Honglak**; The Regents of The University of Michigan, 1600 Huron Parkway, 2nd Floor, Ann Arbor, Michigan 48109-2590 (US). **SOHN, Kihyuk**; The Regents of The University of Michigan, 1600 Huron Parkway, 2nd Floor, Ann Arbor, Michigan 48109-2590 (US).

(74) Agents: **MACINTYRE, Timothy D.** et al.; Harness, Dickey & Pierce, P.L.C., P.O. Box 828, Bloomfield Hills, Michigan 48303 (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,

[Continued on next page]

(54) Title: DEEP LEARNING FRAMEWORK FOR GENERIC OBJECT DETECTION

(57) Abstract: Object detection remains a fundamental problem and bottleneck to be addressed for making vision algorithms practical. Despite the promise, deep learning methods have not been extensively investigated on object detection problems. In this disclosure, deep learning approaches are developed for object detection problems. Specifically, learning algorithms are developed that learn hierarchical features (e.g., object parts) that can provide useful discriminative information for object detection tasks. In addition, algorithms are developed to improve invariance and discriminative power of the learned features.

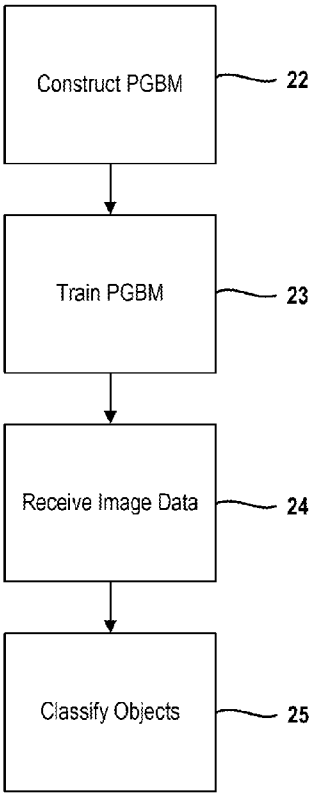


FIG. 2

WO 2014/205231 A1

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, IR, IS, JP, KE, KG, 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, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

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

DEEP LEARNING FRAMEWORK FOR GENERIC OBJECT DETECTION

GOVERNMENT CLAUSE

[0001] This invention was made with government support under IIS1247414 awarded by the National Science Foundation. The government has certain rights in this invention.

CROSS-REFERENCE TO RELATED APPLICATIONS

[0002] This application claims the benefit of U.S. Provisional Application No. 61/836,845 filed on June 19, 2013. The entire disclosure of the above application is incorporated herein by reference.

FIELD

[0001] The present disclosure relates to deep learning framework for generic object detection.

BACKGROUND

[0002] Deep learning has emerged as a promising approach to solve challenging computer vision problems. For example, deep learning and feature learning methods have been successfully applied to object and scene categorization problems. However, object detection still remains a fundamental problem and bottleneck to be addressed for making vision algorithms practical. Despite the promise, deep learning methods have not been extensively investigated on object detection problems. In this disclosure, deep learning approaches are developed for object detection problems. Specifically, learning algorithms are developed that learn hierarchical features (e.g., object parts) that can provide useful discriminative information for object detection tasks. In addition, algorithms are developed to improve invariance and discriminative power of the learned features.

[0003] This section provides background information related to the present disclosure which is not necessarily prior art.

SUMMARY

[0004] This section provides a general summary of the disclosure, and is not a comprehensive disclosure of its full scope or all of its features.

[0005] In one aspect, an automated technique is provided for classifying objects in an image. This technique employs a point-wise gated Boltzmann machine having a visible layer of units, a corresponding switching unit for each of the visible units and at least one hidden layer of units, where the visible units represent intensity values for pixels in an image and the switching units determine which hidden units generate corresponding visible units. The method includes: receiving data for an image captured by an imaging device; and classifying objects in the image data using the point-wise gated Boltzmann machine.

[0006] In another aspect, an automated technique is provided for identifying features in an image. This technique employs a feature recognition model that combines an energy function for a restricted Boltzmann machine with an energy function for conditional random fields. The method includes: receiving data for an image captured by an imaging device; segmenting pixels of the image data into two or more regions using the feature recognition model; and labeling the segmented regions of the image data using the feature recognition model.

[0007] Further areas of applicability will become apparent from the description provided herein. The description and specific examples in this summary are intended for purposes of illustration only and are not intended to limit the scope of the present disclosure.

DRAWINGS

[0008] The drawings described herein are for illustrative purposes only of selected embodiments and not all possible implementations, and are not intended to limit the scope of the present disclosure.

[0009] Figures 1A and 1B are graphical model representations of the point-wise gated Boltzmann machine (PGBM) and supervised PGBM with two groups of hidden units, respectively;

[0010] Figure 2 is a flowchart depicting an automated technique for classifying objects in an image using the point-wise gated Boltzmann machine (PGBM);

[0011] Figures 3A and 3B are visualizations of filters corresponding to two components learned from PGBM;

[0012] Figure 3C is a visualization of the activation of switch units;

[0013] Figure 3D is a visualization of the corresponding original images on *mnist-back-image* dataset;

[0014] Figure 4 is a visualization of the switch unit activation map (top) and images overlayed with the predicted and the ground truth bounding boxes (bottom);

[0015] Figure 5 is a diagram depicting the proposed global and local (GLOC) model;

[0016] Figures 6A and 6B are successful and unsuccessful sample segmentation results on images from the LFW data set;

[0017] Figure 7 illustrates some of the latent structure automatically learned by the GLOC model;

[0018] Figure 8 is a schema for constructing mid-level feature extraction;

[0019] Figures 9A and 9B are factor graphs for the single-task and multi-task, respectively, Beta-Bernoulli process restricted Boltzmann machine;

[0020] Figure 10 is a graph of the area under the ROC curve of each of the 64 attributes for the BBP-RBM features corresponding to labeled attributes (circles) and the attribute classifiers trained using the base features (squares);

[0021] Figure 11 is a schematic illustration of congealing of one dimensional binary images, where the transformation space is left-right translation;

[0022] Figure 12 is diagram depicting a convolutional RBM with probabilistic max-pooling;

[0023] Figures 13A and 13B are visualizations of second layer filters learned from face images without topology and with topology, respectively;

[0024] Figure 14 are sample images from LFW produced by different alignment algorithms;

[0025] Figure 15 is a schematic diagram of convolutional RBM with probabilistic max-pooling;

[0026] Figure 16 is graph showing random filter accuracy versus learned filter accuracy for a one-layer network, using a single image cropping and no metric learning (SVM only);

[0027] Figures 17A and 17B are histrograms over the number of representations correctly classifying each pair, for matched and mismatched pairs, respectively (cut off at 100 pairs);

[0028] Figure 18 is a feature encoding of TIRBM;

[0029] Figures 19A and 19B are translation and scale transformations on images;

[0030] Figures 20A-20D, are samples from the handwritten digit datasets with no transformations, rotation, scaling, and translation, respectively; and Figures 20E and 20F are learned filters from mnist-rot data set with the sparse TIRBM and the sparse RBM, respectively; and

[0031] Figures 21A- 21D are visualization of filters trained with RBM and TIRBMs on natural images.

[0032] Corresponding reference numerals indicate corresponding parts throughout the several views of the drawings.

DETAILED DESCRIPTION

[0033] Example embodiments will now be described more fully with reference to the accompanying drawings.

[0034] One fundamental difficulty in building algorithms that can robustly learn from complex real-world data is to deal with significant noise and irrelevant patterns. In particular, consider a problem of *learning from scratch,* assuming the lack of useful raw features. Here, the challenge is how to learn a robust representation that can distinguish important (e.g., task-relevant) patterns from significant amounts of distracting (e.g., task-irrelevant) patterns.

[0035] For constructing useful features, unsupervised feature learning has emerged as a powerful tool in learning representations from

unlabeled data. In many real-world problems, however, the data is not cleaned up and contains significant amounts of irrelevant sensory patterns. In other words, *not all patterns are equally important*. In this case, the unsupervised learning methods may blindly represent the irrelevant patterns using the majority of the learned high-level features, and it becomes even more difficult to learn task-relevant higher-layer features (e.g., by stacking). Although there are ways to incorporate supervision (e.g., supervised fine-tuning), learning is still challenging when the data contains lots of irrelevant patterns.

[0036] To deal with such complex data, one may envision using feature selection. Indeed, feature selection is an effective method for distinguishing useful raw features from irrelevant raw features. However, feature selection may fail if there are no good raw features to start with.

[0037] To address this issue, this disclosure proposes to combine feature learning and feature selection coherently in a unified framework. Intuitively speaking, given that unsupervised feature learning can find partially useful high-level abstractions, it may be easier to apply feature selection on learned high-level features to distinguish the task-relevant ones from the task-irrelevant ones. Then, the task-relevant high-level features can be used to trace back where such important patterns occur. This information can help the learning algorithm to focus on these task-relevant raw features (i.e., visible units corresponding to task-relevant patterns), while ignoring the rest.

[0038] This disclosure formulates a generative feature learning algorithm called the *point-wise gated Boltzmann machine (PGBM).* The model performs feature selection not only on learned high-level features (i.e., hidden units), but also on raw features (i.e., visible units) through a gating mechanism using stochastic "switch units." The switch units allow the model to estimate where the task-relevant patterns occur, and make only those visible units to contribute to the final prediction through multiplicative interaction. The model ignores the task-irrelevant portion of the raw features, thus it performs *dynamic feature selection* (i.e., choosing a variable subset of raw features depending on semantic interpretation of the individual example).

[0039] The model can be viewed as a high-order extension of the restricted Boltzmann machine (RBM). The RBM is an undirected graphical model

that defines the distribution of visible units using binary hidden units. The joint distribution of binary visible units and binary hidden units is written as follows:

$$P(\mathrm{v},\mathrm{h}) = \frac{1}{Z}\exp\big(-E(\mathrm{v},\mathrm{h})\big),$$
$$E(\mathrm{v},\mathrm{h}) = -\sum_{i=1}^{D}\sum_{k=1}^{K} v_i W_{ik} h_k - \sum_{k=1}^{K} b_k h_k - \sum_{i=1}^{D} c_i v_i$$

where $\boldsymbol{v} \in \{\mathbf{0},\mathbf{1}\}^D$ are the visible (i.e., input) units, and $\boldsymbol{h} \in \{\mathbf{0},\mathbf{1}\}^K$ are the hidden (i.e., latent) units. Z is the normalizing constant, and $\boldsymbol{W} \in \mathbb{R}^{D\times K}$, $\boldsymbol{b} \in \mathbb{R}^K$, $\boldsymbol{c} \in \mathbb{R}^D$ are the weight matrix, hidden and visible bias vectors, respectively. Since there are no connections between the units in the same layer, visible units are conditionally independent given the hidden units, and vice versa. The conditional probabilities of the RBM can be written as follows:

$$P(v_i = 1|\mathrm{h}) = \sigma\left(\sum_k W_{ik} h_k + c_i\right),$$
$$P(h_k = 1|\mathrm{v}) = \sigma\left(\sum_i W_{ik} v_i + b_k\right),$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$. Training the RBM corresponds to maximizing the log-likelihood of the data with respect to parameters **{W, b, c}.** Although the gradient is intractable to compute, contrastive divergence can be used to approximate it.

[0040] A basic unsupervised PGBM that learns and groups features into semantically distinct components is described below. When dealing with complex data, it is desirable for a learning algorithm to distinguish semantically distinct patterns. For example, an object recognition algorithm may improve its performance if it can separate the foreground object patterns from the background clutters. To model this, each visible unit is represented as a mixture model when conditioned on the hidden units, where each group of hidden units can generate the corresponding mixture component.

[0041] Before going into details, the generative process of the PGBM is described as follows: (1) the hidden units are partitioned into *components*, each of which defines a distinct distribution over the visible units. (2) Conditioning on the hidden units, sample the *switch units*. (3) The switch units determine which *component* generates the corresponding visible units. A schematic diagram of a PGBM is shown in Figure 1A as an undirected graphical model.

[0042] The PGBM with R mixture components has a multinomial switch unit, denoted $z_i \in \{1, \ldots, R\}$[1] for each visible unit v_i. The PGBM imposes element-wise multiplicative interaction between the paired switch and visible units, as shown in Figure 1A. Now, the energy function of the PGBM is defined as follows:

$$\begin{aligned} E^U(\mathrm{v}, \mathrm{z}, \mathrm{h}) = & -\sum_{r=1}^{R}\sum_{i=1}^{D}\sum_{k=1}^{K_r}(z_i^r v_i)W_{ik}^r h_k^r \\ & -\sum_{r=1}^{R}\sum_{k=1}^{K_r} b_k^r h_k^t - \sum_{r=1}^{R}\sum_{i=1}^{D}(z_i^r v_i)c_i^r, \\ s.t. & \quad \sum_{r=1}^{R} z_i^r = 1, i = 1, \ldots, D. \end{aligned} \tag{1}$$

Here, v, z^r and h are the visible, switch and hidden unit binary vectors, respectively, and the model parameters W_{ik}^r, b_k^r, c_i^r are the weights, hidden biases, and the visible biases of r-th component. The binary-valued switch unit z_i^r is activated (i.e. takes value 1) if and only if its paired visible unit v_i is assigned to the r-th component, and its conditional probability given hidden units follows a multinomial distribution over R categories. The energy function can be written in matrix form as follows:

$$\begin{aligned} E^U(\mathrm{v}, \mathrm{z}, \mathrm{h}) = & -\sum_{r=1}^{R}(\mathrm{z}^r \odot \mathrm{v})^T W^r h^r \\ & -\sum_{r=1}^{R}(b^r)^T h^r - \sum_{r=1}^{R}(c^r)^T(z^r \odot \mathrm{v}), \end{aligned}$$

where the operation $\odot$ denotes element-wise multiplication, i.e., $(z^r \odot v)_i = z_i^r v_i$.

[0043] The visible, hidden, and switch units are conditionally independent given the other two types of units, and the conditional probabilities can be written as follows:

$$P(h_k^r = 1 \mid \mathbf{z}, \mathbf{v}) = \sigma\left((\mathbf{z}^r \odot \mathbf{v})^T \mathbf{W}_{:k}^r + b_k^r\right), \tag{2}$$

$$P(v_i = 1 \mid \mathbf{z}, \mathbf{h}) = \sigma\left(\sum_r z_i^r (\mathbf{W}_{i:}^r \mathbf{h}^r + c_i^r)\right), \tag{3}$$

$$P(z_i^r = 1 \mid \mathbf{v}, \mathbf{h}) = \frac{\exp\left(v_i (\mathbf{W}_{i:}^r \mathbf{h}^r + c_i^r)\right)}{\sum_s \exp\left(v_i (\mathbf{W}_{i:}^s \mathbf{h}^s + c_i^s)\right)}, \tag{4}$$

where W_i^r, is used to denote i-th row, and $W_{:k}^r$ is used to denote k-th column of the matrix W^r.

[0044] It is important to note that, while inferring the hidden units, the model gates (or re-weighs) each visible unit v_i according to the corresponding switch units z_i^r (Equation 2). In other words, the point-wise multiplicative interaction between the switch and the visible units allows the hidden units in each component to focus on a specific part of the data, and this makes the hidden units in one component to be robust to the patterns learned by other components. Moreover, the top-down signal from the hidden units encourages assigning the same mixture component to semantically-related visible units during the switch unit inference, and therefore we can prune out the irrelevant raw features dynamically for each example.

[0045] It is worth noting that, when all switch units (i.e., $z_i = z \text{ for all } i$) are tied, the PGBM becomes equivalent to the implicit mixture of restricted Boltzmann machine. Furthermore, given that there is a multiplicative interaction between three types of variables, the PGBM can be understood in the context of higher-order Boltzmann machines.

[0046] The PGBM is trained with stochastic gradient descent using contrastive divergence. Since the exact inference is intractable due to the three-way interaction, mean-field or alternating Gibbs sampling (i.e., sample one type of variables given the other two types using Equations (2),(3), and (4)) is used for approximate inference.

[0047] Although the PGBM can learn to group distinct features for each mixture component, it doesn't necessarily learn discriminative features automatically since the generative training is done in an unsupervised way. One way to make the PGBM implicitly perform feature selection (i.e., distinguish features into different groups based on their relevance to the task) is to provide a good initialization of the model parameters. For example, pre-train the regular RBM and divide the hidden units into two groups based on the score from the simple feature selection algorithms, such as the t-test, to initialize the weight matrices of the PGBM. As further discussed below, this approach improves classification performance of the PGBMs.

[0048] Furthermore, to make use of class labels during the generative training, a supervised PGBM is proposed that only connects the hidden units in the task-relevant component(s) to the label units. The graphical

model representation is shown in Figure 1B. By transferring the label information to the raw features through the task-relevant hidden units, the supervised PGBM can perform generative feature selection both at the high-level (i.e., using only a subset of hidden units for classification) and the low-level (e.g., dynamically blocking the influence of the task-irrelevant visible units) in a unified way.

[0049] For simplicity, the supervised PGBM is presented with two mixture components, where the first component is assigned to be task-relevant. The energy function is defined as follows:

$$E^s(\mathrm{v},\mathrm{z},\mathrm{h},\mathrm{y}) = E^U(\mathrm{v},\mathrm{z},\mathrm{h}) - y^T U h^1 - d^T y \quad (5)$$

subject $to\ z_i^1 + z_i^2 = 1,\ i-1,\ldots,D$. The label vector $y \in \{0,1\}^L$ is in the 1-of-L representation. $U \in \mathbb{R}^{L\times K_1}$ is the weight matrix between the task-relevant hidden units and the label units, and d is the label bias vector. The conditional probabilities can be written as follows:

$$P(h_k^1 = 1|\mathrm{z},\mathrm{v},\mathrm{y}) = \sigma((z^1 \odot \mathrm{v})^T W_{.k}^1 + b_k^1 + U_{.k}^T), \quad (6)$$

$$P(y_l = 1|\mathrm{h}^1) = \frac{\exp(\mathrm{U}_L \mathrm{h}^1 + d_l)}{\sum_s \exp(\mathrm{U}_S \mathrm{h}^1 + d_s)}. \quad (7)$$

The conditional probabilities of the visible and switch units are the same as Equations (3) and (4). As seen in Equation (6), the label information, together with the switch units, modulates the hidden unit activations in the first (task-relevant) component, and this in turn encourages the switch units z_i^1 to activate at the task-relevant visible units during the iterative approximate inference. In the model, a visible unit (a raw feature) is "task-relevant" if its switch unit for the task-relevant component is active, respectively.

[0050] The supervised PGBM can be trained in generative criteria whose objective is to maximize the joint log-likelihood of the visible and the label units. Similarly to that of PGBM, the inference can be done with alternative Gibbs sampling between Equations (3), (4), (6), and (7).

[0051] Figure 2 depicts an example automated technique for classifying objects in an image using the point-wise gated Boltzmann machine (PGBM) set forth above. The PGBM is first constructed at 22. In the example embodiment, the PGBM includes a visible layer of units, a corresponding

switching unit for each of the visible units and at least one hidden layer of units, where the visible units represent intensity values for pixels in an image and the switching units determine which hidden units generate the corresponding visible units. In some instances, the PGBM is implemented by computer-readable instructions residing in a non-transitory data store and executed by a computer processor.

[0052] Prior to employing the PGBM, it is preferably trained at 23 using one of the methods described above. The PGBM can then be used at 25 to classify objects, for example contained in image data captured by an imaging device, such as a camera. There are many classification tasks where a large number of unlabeled examples are provided in addition to only a small number of labeled training examples. For this scenario, it is important to include unlabeled examples during training to generalize well to the unseen data. In other embodiments, the supervised PGBM can be adapted to the semi-supervised learning framework. For example, the joint log-likelihood $\log P^S(v, y)$ can be regularized with the data log-likelihood $\log P^S(v)$ defined on the unlabeled data.

[0053] In yet other embodiments, the PGBM can be used as a building block of deep networks. For example, PGBM can be used as a first layer block and stack neural networks on the hidden units of task-relevant components of the PGBM. Since the PGBM can select the task-relevant hidden units with supervision, the higher-layer networks can focus on the task-relevant information. Below it is shown that the two-layer model, where a single-layer neural network is stacked on top of a PGBM's task-relevant component, was sufficient to outperform existing state-of-the-art classification performance on the variations of MNIST dataset with irrelevant backgrounds.

[0054] Convolutional models can be useful in representing spatially or temporally correlated data. The PGBM can be extended to a convolutional setting, where the filter weights are shared over different locations in large images. Below the convolutional PGBM is presented with an application to the weakly supervised foreground object localization problem. Furthermore, by locating the bounding box at the foreground object accurately, state-of-the-art recognition performance is achieved in Caltech 101.

[0055] Next, the capability of the proposed models in learning task-relevant features from noisy data is evaluated. The single-layer PGBMs and their extensions are tested on the variations of MNIST dataset: *mnist-back-rand, mnist-back-image, mnist-rot-back-image,* and *mnist-rot-back-rand.* The first two datasets use uniform noise or natural images as background patterns. The other two have rotated digits in front of the corresponding background patterns. The PGBM with two components of 500 hidden units is used and initialized with the pre-trained RBM using the feature selection as described above. Mean-field is used for approximate inference for these experiments.

[0056] With reference to Figure 3, the filters and the switch unit activations are visualized for *mnist-back-image.* The foreground filters capture the task-relevant patterns resembling pen strokes (Figure 3A), while the background filters capture task-irrelevant patterns in the natural images (Figure 3B). Further, the switch unit activations (the posterior probabilities that the input pixel belongs to the *foreground component,* Figure 3C) are high (colored in white) for the foreground digit pixels, and low (colored in gray) for the background pixels. This suggests that the model can *dynamically* separate the task-relevant raw features from the task-irrelevant raw features for each example.

[0057] For quantitative evaluation, test classification errors are enumerated in Table 1 below. For all experiments with single-layer models, the "task-relevant" hidden unit activations are used as the input for the linear SVM. The single-layer PGBM significantly outperformed the baseline RBM, imRBM, and discRBM. A careful model selection was done to choose the best hyperparameters for each of the compared models. These results suggest that the point-wise mixture hypothesis is effective in learning task-relevant features from complex data containing irrelevant patterns.

Algorithm	Mnist-back-rand	Mnist-back-image	Mnist-rot-back-image	Mnist-rot-back-rand
RBM	11.39	15.42	49.89	51.97
imRBM	10.46	16.35	51.03	51.02
discRBM	10.29	15.56	48.34	48.28
RBM-FS	11.42	15.20	49.65	51.69
PGBM	7.27	13.33	45.45	45.53
supervised PGBM	6.87	12.85	44.67	43.47
DBN-3 (Vincent et al., 2008)	6.73	16.31	47.39	-
CAE-2(Rifai et al., 2011)	10.90	15.50	45.23	-
PGBM+ DN-1	6.08	12.25	36.76	30.41

[0058] As a control experiment, the model was compared to the two-step model which is referred to herein as "RBM-FS", where we first trained the RBM and selected a subset of hidden units using feature selection. As seen in Table 1, the RBM-FS is only marginally better (or sometimes worse) than the baseline RBM. However, the PGBM significantly outperforms the RBM-FS, which demonstrates the benefit of the joint training.

[0059] The supervised PGBM can be trained in a semi-supervised way as described above. The same experimental setting was used as described by Larochelle and Bengio in "Classification using discriminative restricted Boltzmann machines" In ICML, 2008, and provided labels for only 10 percent of training examples (100 labeled examples for each digit category). The classification errors of semi-supervised PGBM, supervised PGBM, RBM and RBM-FS are summarized in Table 2 below. The semi-supervised PGBM consistently performed the best for all datasets, showing that semi-supervised training is effective in utilizing a large number of unlabeled examples.

Algorithm	mnist-back-rand	mnist-back-image	mnist-rot-back-image	mnist-rot-back-rand
RBM	17.43 ± 0.36	23.71 ± 0.34	63.94 ± 0.50	63.17 ± 0.32
RBM-FS	17.15 ± 0.46	20.22 ± 0.31	61.76 ± 0.43	62.02 ± 0.81
supervised PGBM	16.15 ± 0.70	21.04 ± 0.18	59.39 ± 0.58	63.82 ± 0.68
semi-supervised PGBM	11.98 ± 0.80	20.32 ± 0.15	59.19 ± 0.68	58.57 ± 0.49

[0060] Finally, a two-layer deep network was constructed by stacking one layer of neural network with 1,000 hidden units on the task-relevant component of the PGBM. A softmax classifier was used for fine-tuning of the second layer neural network. Table 1 shows that the deep network (referred to herein as "PGBM+DN-1") outperforms the DBN-3 and the stacked contractive autoencoder by a large margin. In particular, the result of the DBN-3 on *mnist-back-image* implies that adding more layers to the DBN does not necessarily improve the performance when there are significant amounts of irrelevant patterns in the data. In contrast, the PGBM can block the task-irrelevant information from propagating to the higher layers, and hence it is an effective building block for deep networks. It is noted that the PGBM+DN-1 achieved state-of-the-art classification performance on all datasets except *mnist-rot-back-image,* where the transformation-invariant RBM achieved 35.5% error by incorporating the rotational invariance.

[0061] The model can be extended to learn groups of task-relevant features (i.e., foreground patterns) from the images with higher resolution, and apply it to weakly supervised object segmentation.

[0062] To learn and group related features from large images, a point-wise gated convolutional deep network (CPGDN) is proposed, where we use the convolutional extension of the PGBM (CPGBM) as a building block. Specifically, the two-year CPGDN is constructed by stacking the CPGBM on the first layer CRBM. This construction makes sense because the first layer features are mostly generic, and the class-specific features emerge in higher layers. The CPGDN is trained using greedy layer-wise training method, and feedforward inference is performed in the first layer. Mean-field is used in the second layer for approximate inference of switch and hidden units.

[0063] First, a CPGDN is trained with two mixture components only on the single class of images from Cal-tech 101 dataset. For this experiment, the weights are randomly initialized without pre-training. The second layer features are trained on "Faces" and "Car side" classes. The CPGDN made a good distinction between the task-relevant patterns such as face parts and wheels, and the generic patterns. In Figure 4, the switch unit activation map is visualized, which shows that the switch units are selectively activated at the most informative region in each image. Interestingly, using this activation map, the object region can be segmented from the background reasonably well, though the model is not specifically designed for image segmentation.

[0064] Inspired by the CPGDN's ability to distinguish the foreground object from the background scene, a novel object recognition pipeline on Caltech 101 dataset is proposed, where each image is first “crop” at the bounding box predicted using the switch unit activations of the CPGDN and classification is performed using those cropped images. Specifically, the CPGDN is used with two mixture components, each of which is composed of 100 hidden units. To train the model efficiently from many different classes of images, a set of second layer CRBMs is pre-trained with a small number of hidden units (e.g., 30) for each class to capture more diverse and class-specific patterns, and perform feature selection on those CRBM features from all object categories to initialize the weights of the second layer CPGBM. Once the model is trained, the

posterior of switch units arranged in 2d is computed. To predict the bounding box, compute the row-wise and column-wise cumulative sum of switch unit activations and select the region containing (5,95) percentiles of the total activations as a bounding box. For classification, follow the pipeline used in Sohn, et al's "Efficient Learning of Sparse, Distributed, Convolutional Feature Representations For Object Recognition", ICCV, 2011, which uses the Gaussian (convolutional) RBMs with dense SIFT as input.

[0065] First, the bounding box detection accuracy is evaluated. The bounding box prediction is declared correct when the average overlap ratio (the area of intersection divided by the union between the predicted and the ground truth bounding boxes) is greater than 0.5. Average overlap ratio of 0.702 and detection accuracy of 88.3% is achieved.

[0066] Finally, the classification accuracy is evaluated using the cropped Caltech 101 dataset with CPGDN and summarize the results in Table 3. The object centered cropped images brought improvement in classification accuracies, such as 74.9% to 76.8% with RBM, and 77.8% to 78.9% with CRBM using 30 training images per class, respectively. As a baseline, the classification accuracy on the augmented dataset is reported where uniformly crop the center region across all the images with a fixed ratio. After cross-validating with different ratios, a worse classification accuracy of 75.8% is obtained with RBM using 30 training images per class. This suggests that the classification performance can be improved by localizing the object better than simply cropping the center region.

Training images per class	15	30
Lazebnik et al. (2006)	56.4%	64.6%
Griffin et al. (2007)	59.0%	67.6%
Yang et al. (2009)	67.0%	73.2%
Boureau et al. (2010)	-	75.7%
Goh et al. (2012)	71.1%	78.9%
RBM (Sohn et al., 2011)	68.6%	74.9%
Our method + RBM	70.2%	76.8%
CRBM (Sohn et al., 2011)	71.3%	77.8%
Our method + CRBM	72.4%	78.9%

[0067] Segmentation and region labeling are core techniques for the critical mid-level vision tasks of grouping and organizing image regions into coherent parts. Segmentation refers to the grouping of image pixels into parts

without applying labels to those parts, and region labeling assigns specific category names to those parts. While many segmentation and region labeling algorithms have been used in general object recognition and scene analysis, they have played a surprisingly small role in the challenging problems of face recognition.

[0068] In another aspect of this disclosure, the problem of labeling face regions with hair, skin is addressed, and background labels as an intermediate step in modeling face structure. In region labeling applications, the conditional random field (CRF) is effective at modeling region boundaries. For example, the CRF can make a correct transition between the hair and background labels when there is a clear difference between those regions. However, when a person's hair color is similar to that of the background, the CRF may have difficulty deciding where to draw the boundary between the regions. In such cases, a global shape constraint can be used to filter out unrealistic label configurations.

[0069] It has been shown that restricted Boltzmann machines (RBMs) and their extension to deeper architectures such as deep Boltzmann machines (DBMs), can be used to build effective generative models of object shape. Specifically, the recently proposed shape Boltzmann machine (ShapeBM) showed impressive performance in generating novel but realistic object shapes while capturing both local and global elements of shape.

[0070] Motivated by these examples, a *GLOC* (GLObal and LOCal) model is proposed; a strong model for image labeling problems, that combines the best properties of the CRF (that enforces *local consistency* between adjacent nodes) and the RBM (that models *global shape prior* of the object). The model balances three goals in seeking label assignments: the region labels should be consistent with the underlying image features; the region labels should respect image boundaries; and the complete image labeling should be consistent with shape priors defined by the segmentation training data. In the GLOC model, the first two objectives are achieved primarily by the CRF part, and the third objective is addressed by the RBM part. For each new image, the model uses mean-field

inference to find a good balance between the CRF and RBM potentials in setting the image labels and hidden node values.

[0071] CRF and RBM are described followed by the proposed GLOC model (also referred to herein as feature recognition model). The models are presented in the context of multi-class labeling. An image I is pre-segmented into $S^{(I)}$ superpixels, where $S^{(I)}$ can vary over different images. Denote $\mathcal{V}^{(I)} = \{1, \ldots, S^{(I)}\}$ as a set of superpixel nodes, and $\mathcal{E}^{(I)}$ as a set of edges connecting adjacent superpixels. Denote $\mathcal{X}^{(I)} = \{\mathcal{X}_{\mathcal{V}}^{(I)}, \mathcal{X}_{\mathcal{E}}^{(I)}\}$, where $\mathcal{X}_{\mathcal{V}}^{(I)}$ is a set of node features $\{x_s^{node} \in \mathbb{R}^{D_n}, s \in \mathcal{V}\}$ and $\mathcal{X}_{\mathcal{E}}^{(I)}$ is a set of edge features $\{x_{ij}^{edge} \in \mathbb{R}^{D_e}, (i,j) \in \mathcal{E}\}$. The set of label nodes are defined as $\mathcal{Y}^{(I)} = \{y_s \in \{0,1\}^L, s \in \mathcal{V} : \sum_{l=1}^{L} y_{sl} = 1\}$. Here, D_n and D_e denote the dimensions of the node and edge features, respectively, and L denotes the number of categories for the labeling task. The superscripts "I" "node", or "edge" are frequently omitted for clarity, but the meaning should be clear from the context.

[0072] The conditional random field is a powerful model for structured output prediction (such as sequence prediction, text parsing, and image segmentation), and has been used in computer vision. The conditional distribution and the energy function can be defined as follows:

$$P_{crf}(\mathcal{Y}|\mathcal{X}) \propto \exp(-E_{crf}(\mathcal{Y}, \mathcal{X})), \quad (1)$$

$$E_{crf}(\mathcal{Y}, \mathcal{X}) = E_{node}(\mathcal{Y}, \mathcal{X}_{\mathcal{V}}) + E_{edge}(\mathcal{Y}, \mathcal{X}_{\mathcal{E}}), \quad (2)$$

$$E_{node}(\mathcal{Y}, \mathcal{X}_{\mathcal{V}}) = -\sum_{s \in \mathcal{V}} \sum_{l=1}^{L} \sum_{d=1}^{D_n} y_{sl} \Gamma_{ld} x_{sd},$$

$$E_{edge}(\mathcal{Y}, \mathcal{X}_{\mathcal{E}}) = -\sum_{(i,j) \in \mathcal{E}} \sum_{l,l'=1}^{L} \sum_{e=1}^{D_e} y_{il} y_{jl'} \Psi_{ll'e} x_{ije},$$

where $\Psi \in \mathbb{R}^{L \times L \times D_e}$ is a 3D tensor for the edge weights, and $\Gamma \in \mathbb{R}^{L \times D_n}$ are the node weights. The model parameters $\{\Gamma, \Psi\}$ are trained to maximize the conditional log-likelihood of the training data $\{\mathcal{Y}^{(m)}, \mathcal{X}^{(m)}\}_{m=1}^{M}$,

$$\max_{\Gamma,\psi} \sum_{m=1}^{M} \log P_{\mathrm{crf}(y^{(m)}|x^{(m)})}.$$

Loopy belief propagation (LBP) or mean-field approximation can be used for inference in conjunction with standard optimization methods such as LBFGS.

[0073] The restricted Boltzmann machine is a bipartite, undirected graphical model composed of visible and hidden layers. In this context, assuming R^2 multinomial visible units $y_r \in \{0,1\}^L$ and K binary hidden units $h_k \in \{0,1\}$, the joint distribution can be defined as follows:

$$P_{\mathrm{rbm}}(\mathcal{Y}, \mathrm{h}) \propto \exp\left(-E_{\mathrm{rbm}}(\mathcal{Y}, \mathrm{h})\right), \tag{3}$$

$$\begin{aligned} E_{\mathrm{rbm}}(\mathcal{Y}, \mathrm{h}) = &-\sum_{r=1}^{R^2}\sum_{l=1}^{L}\sum_{k=1}^{K} y_{rl} W_{rlk} h_k \\ &-\sum_{k=1}^{K} b_k h_k - \sum_{r=1}^{R^2}\sum_{l=1}^{L} c_{rl} y_{rl}, \end{aligned} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{R^2 \times L \times K}$ is a 3D tensor specifying the connection weights between visible and hidden units, b_k is the hidden bias, and c_{rl} is the visible bias. The parameters $\Theta = \{\mathbf{W}, \mathrm{b}, \mathbf{C}\}$ are trained to maximize the log-likelihood of the training data $\{\mathcal{Y}^{(m)}\}_{m=1}^{M}$,

$$\max_{\Theta} \sum_{m=1}^{M} \log\left(\sum_{\mathrm{h}} P_{\mathrm{rbm}}(\mathcal{Y}^{(m)}, \mathrm{h})\right).$$

The model parameters are trained using stochastic gradient descent. Although the exact gradient is intractable to compute, it can be approximated using contrastive divergence. Other training methods are also contemplated by this disclosure.

[0074] To build a strong model for image labeling, both local consistency (adjacent nodes are likely to have similar labels) and global consistency (the overall shape of the object should look realistic) are desirable. On one hand, the CRF is powerful in modeling local consistency via edge potentials. On the other hand, the RBM is good at capturing global shape structure through the hidden units. These two ideas are combined in the GLOC

model, which incorporates both local consistency (via CRF-like potentials) and global consistency (via RBM-like potentials). Specifically, the conditional likelihood of labels set $\mathcal{Y}$ given the superpixel features $\mathcal{X}$ is described as follows:

$$P_{gloc}(\mathcal{Y}|\mathcal{X}) \propto \sum_h \exp\left(-E_{gloc}(\mathcal{Y},\mathcal{X},h)\right) \quad (5)$$

$$E_{gloc}(\mathcal{Y},\mathcal{X},h) = E_{crf}(\mathcal{Y},\mathcal{X}) + E_{rbm}(\mathcal{Y},h). \quad (6)$$

As described above, the energy function is written as a combination of CRF and RBM energy functions. However, due to the varying number of superpixels for different images, the RBM energy function in Equation (4) requires nontrivial modifications. In other words, one cannot simply connect label (visible) nodes defined over superpixels to hidden nodes as in Equation (4) because the RBM is defined on a fixed number of visible nodes and the number of superpixels and their underlying graph structure can vary across images.

[0075] To resolve this issue, a *virtual, fixed-sized* pooling layer is introduced between the label layer and the hidden layer, where each superpixel label node is mapped into the *virtual* visible nodes of the $R \times R$ square grid. This is shown in Figure 5, where the top two layers can be thought of as an RBM with the visible nodes $\bar{y}_r$, representing a surrogate (i.e., pooling) for the labels y_s that overlap with the grid bin r. Specifically, the energy function between the label nodes and the hidden nodes for an image I is defined as follows:

$$\begin{aligned} E_{rbm}(\mathcal{Y},h) = &-\sum_{r=1}^{R^2}\sum_{l=1}^{L}\sum_{k=1}^{K}\bar{y}_{rl}W_{rlk}h_k, \\ &-\sum_{k=1}^{K}b_k h_k - \sum_{r=1}^{R^2}\sum_{l=1}^{L}c_{rl}\bar{y}_{rl}. \end{aligned} \quad (7)$$

Here, the virtual visible nodes $\bar{y}_{rl} = \sum_{s=1}^{S} p_{rs}y_{sl}$ are deterministically mapped from the superpixel label nodes using the projection matrix $\{p_{rs}\}$ that determines the contribution of label nodes to each node of the grid. The projection matrix is defined as follows:

$$P_{rs} = \frac{Area(Region(s) \cap Region(r))}{Area(Region(r))},$$

where $Region(s)$ and $Region(r)$ denote sets of pixels corresponding to superpixel s and grid r, respectively. The projection matrix $\{p_{rs}\}$ is a sparse, non-negative

matrix of dimension $R^2 \times S$. Note that the projection matrix is specific to each image since it depends on the structure of the superpixel graph. Due to the deterministic connection, the pooling layer is actually a *virtual* layer that only exists to map between the superpixel nodes and the hidden nodes. The GLOC model can also be viewed as having a set of grid-structured nodes that performs average pooling over the adjacent superpixel nodes.

[0076] As an additional baseline, a modification to the CRF is described. In some cases, even after conditioning on class, feature likelihoods may depend on position. For example, knowing that hair rests on the shoulders makes it less likely to be gray. This intuition is behind our Spatial CRF model.

[0077] Specifically, when the object in the image is aligned, a spatially dependent set of weights can be learned that are specific to a cell in an $N \times N$ grid. Note that this grid can be a different size than the $R \times R$ grid used by the RBM. A separate set of node weights for each cell in a grid is learned, but the edge weights are kept globally stationary.

[0078] Using a similar projection technique to that described above, the node energy function is defined as

$$E_{node}(\mathcal{Y}, \mathcal{X}_v) = -\sum_{s \in v} \sum_{l=1}^{L} y_{sl} \sum_{n=1}^{N^2} p_{sn} \sum_{d=1}^{D_n} \Gamma_{ndl} x_{sd}, \quad (8)$$

where $\Gamma \in \mathbb{R}^{N^2 \times D \times L}$ is a 3D tensor specifying the connection weights between the superpixel node features and labels at each spatial location. In this energy function, a different projection matrix $\{p_{sn}\}$ is defined which specifies the mapping from the $N \times N$ virtual grid to superpixel label nodes. Note that the projection matrices used in the RBM and spatial CRF are different in that $\{p_{rs}\}$ used in the RBM describes a projection from superpixel to grid $(\sum_{s=1}^{S} p_{rs} = 1)$; whereas $\{p_{sn}\}$ used in the spatial CRF describes a mapping from a grid to superpixel $\left(\sum_{n=1}^{N^2} p_{sn} = 1\right)$.

[0079] Since the joint inference of superpixel labels and the hidden nodes is intractable, mean-field approximation is employed. Specifically, find a fully factorized distribution $Q(\mathcal{Y}, \mathbf{h}; \mu, \gamma) = \Pi_{s \in v} Q(y_s) \Pi_{k=1}^{K} Q(h_k)$, with $Q(y_s = l) \triangleq \mu_{sl}$ and $Q(h_k = 1) \triangleq \gamma_k$, that minimizes $\mathrm{KL}\left(Q(\mathcal{Y}, \mathbf{h}; \mu, \gamma) \parallel P(\mathcal{Y}, \mathbf{h} | \mathcal{X})\right)$. The mean-field inference steps in set forth below in Algorithm 1.

Algorithm 1 Mean-Field Inference

1: Initialize $\mu^{(0)}$ and $\gamma^{(0)}$ as follows:

$$\mu_{sl}^{(0)} = \frac{exp(f_{sl}^{node})}{\sum_{l'} \exp f_{sl'}^{node}}$$

$$\gamma_k^{(0)} = sigmoid\left(\sum_{r,l}\left(\sum_s p_{rs}\mu_{sl}^{(0)}\right)W_{rlk} + b_k\right)$$

where

$$f_{sl}^{node}(\mathcal{X}_v\{p_{sn}\}, \Gamma) = \sum_{n,d} p_{sn} x_{sd} \Gamma_{ndl}$$

2: for t=0:*maxiter* (or until convergence) **do**

3: update $\mu^{(t+1)}$ as follows: $\mu_{sl}^{(t+1)} =$

$$\frac{exp\left(f_{sl}^{node} + f_{sl}^{edge}(\mu^{(t)}) + f_{sl}^{rbm}(\Upsilon^{(t)})\right)}{\Sigma_{l'} \exp\left(f_{sl'}^{node} + f_{sl'}^{edge}(\mu^{(t)}) + f_{sl'}^{rbm}(\Upsilon^{(t)})\right)}$$

where

$$f_{sl}^{edge}(\mu, \mathcal{X}_\mathcal{E}, \mathcal{E}, \Psi) = \sum_{j:(s,j)\in\mathcal{E}} \sum_{l',e} \mu_{jl'} \Psi_{ll'e} x_{sje}$$

$$f_{sl}^{rbm}(\Upsilon; \{p_{rs}\}, \mathrm{W}, \mathrm{C}) \sum_{r,k} p_{rs}(W_{rlkYk} + c_{rl})$$

4: update $\Upsilon^{(t+1)}$ as follows:

$$\Upsilon_k^{(t+1)} = sigmoid\left(\sum_{r,l}\left(\sum_s p_{rs}\mu_{sl}^{(t+1)}\right)W_{rlk} + b_k\right)$$

5: **end for**

[0080] In principle, the model parameters $\{\mathbf{W}, \mathbf{b}, \mathbf{C}, \Gamma, \Psi\}$ are trained simultaneously to maximize the conditional log-likelihood. In practice, however, it is beneficial to provide a proper initialization (or *pretrain)* to those parameters. An overview of an exemplary training procedure is set forth below in Algorithm 2.

Algorithm 2 Training GLOC model

1: Pretrain $\{\Gamma, \Psi\}$ to maximize the conditional log-likelihood of the $spatial\ CRF$ model (See Equations (1), (2), and (8)).

2: Pretrain $\Theta = \{\mathrm{W}, \mathrm{b}, \mathrm{C}\}$ to maximize the conditional log-liklihood $\log \sum_h P_{crbm}(\mathcal{Y}, h|\mathcal{X}_v)$ of the *conditional RBM* model which is defined as:

$$P_{crbm}(\mathcal{Y}, h|\mathcal{X}_v) \propto \exp\left(-E_{node}(\mathcal{Y}, \mathcal{X}_v; \Gamma) - E_{rbm}(\mathcal{Y}, h; \Theta)\right)$$

3: Train $\{\mathrm{W}, \mathrm{b}, \mathrm{C}, \Gamma, \Psi\}$ to maximize the conditional log-likelihood of the *GLOC* model (See Equation (5)).

[0081] First, the pretraining method of deep Boltzmann machines (DBM) is adapted to train the conditional RBM (CRBM). Specifically, pretrain the model parameters $\{\mathbf{W}, \mathbf{b}, \mathbf{C}\}$ of the CRBM as if it is a top layer of the DBM to avoid double-counting when combined with the edge potential in the GLOC model. Second, the CRBM and the GLOC models can be trained to either maximize the conditional log-likelihood using contrastive divergence (CD) or *minimize generalized perceptron loss* using CD-PercLoss. It was empirically observed that CD-PercLoss performed slightly better than CD.

[0082] In many cases, it is advantageous to learn generative models with deep architectures. In particular, ShapeBM, a special instance of the DBM, can be a better generative model than the RBM when they are only given several hundred training examples. However, when given sufficient training data (e.g., a few thousand), it was found that the RBM can still learn a global shape prior with good generalization performance. The generated samples are diverse and are clearly different from their most similar examples in the training set. This suggests that our model is learning an interesting decomposition of the shape distributions for faces. Furthermore, RBMs are easier to train than DBMs in general, which motivates the use of RBMs in this model. In principle, however, such deep architectures can be used in the GLOC model as a rich global shape prior without much modification to inference and learning.

[0083] The proposed model was evaluated on a task to label face images from the LFW data set as hair, skin, and background. The "funneled" version of LFW was used, in which images have been coarsely aligned using a congealing-style joint alignment approach. Although some better automatic alignments of these images exist, such as the LFW-a data set, LFW-a does not contain color information, which is important for this application.

[0084] The LFW website provides the segmentation of each image into superpixels, which are small, relatively uniform pixel groupings (e.g., available at http://vis-www.cs.umass.edu/lfw/lfw_funneled_superpixels_fine.tgz). Ground truth for a set of 2927 LFW images is provided by labeling each superpixel as either hair, skin, or background. While some superpixels may contain pixels from more than one region, most superpixels are generally "pure" hair, skin, or background.

[0085] There are several reasons why superpixel labeling is used instead of pixel labeling for this problem. First, the superpixel representation is computationally much more efficient. The number of nodes would be too large for pixel labeling since the LFW images are of size 250 x 250. However, each image can be segmented into 200-250 superpixels, resulting in the same number of nodes in the CRF, and this allowed tractable inference using LBP or mean-field. In addition, superpixels can help smooth features such as color. For example, if the superpixel is mostly black but contains a few blue pixels, the blue pixels will be smoothed out from the feature vector, which can simplify inference.

[0086] For each superpixel, the following node features were used:

Color: Normalized histogram over 64 bins generated by running K-means over pixels in LAB space.

Texture: Normalized histogram over 64 texton which are generated according to J. Malik et al's in "Textons, Contours and Regions: Cue Integration In Image Segmentation", ICCV, 1999.

Position: Normalized histogram of the proportion of a superpixel that falls within each of the 8 x 8 grid elements on the image; note that the position feature is only used in the CRF.

The following edge features were computed between adjacent superpixels:

Sum of PB values along the border.

Euclidean distance between mean color histograms.

Chi-squared distance between texture histograms as computed by G.B. Huang et al's in "Towards Unconstrained Face Recognition", CVPR Workshop on Perceptual Organization in Computer Vision, 2008.

The labeling performance is evaluated for four different models: a standard CRF, the spatial CRF, the CRBM, and our GLOC model with the summary results in Table 4. The labeled examples were divided into training, validation, and testing sets that contain 1500, 500, and 927 examples, respectively. The model was trained using batch gradient descent and selected the model hyperparameters that performed best on the validation set. After cross-validation, set $K = 400$, $R = 24$, and $N = 16$. Finally, evaluate that model on the test set. On a multicore AMD Opteron, average inference time per example was 0.254 seconds for the GLOC model and 0.063 seconds for the spatial CRF.

Table 4

Method	Accuracy (SP)	Error Reduction
CRF	93.23%	-
Spatial CRF	93.95%	10.64%
CRBM	94.10%	12.85%
GLOC	94.95%	25.41%

As shown in Table 4, the GLOC model substantially improves the superpixel labeling accuracy over the baseline CRF model as well as the spatial CRF and CRBM models. While absolute accuracy improvements (necessarily) become small as accuracy approaches 95%, the reduction in errors are substantial.

[0087] Furthermore, there are significant qualitative differences in many cases, as illustrated in Figure 6A. The samples on the left show significant improvement over the spatial CRF, and the ones on the right show more subtle changes made by the GLOC model. Here, the confidence of the guess (posterior) is represented by color intensity. The confident guess appears as a strong red, green, or blue color, and a less confident guess appears as a lighter mixture of colors. As seen, the global shape prior of the GLOC model helps "clean up" the guess made by die spatial CRF in many cases, resulting in a more confident prediction.

[0088] In many cases, the RBM prior encourages a more realistic segmentation by either "filling in" or removing parts of the hair or face shape. For example, the woman in the second row on the left set recovers the left side of her hair and gets a more recognizable hair shape under the proposed model. Also, the man in the first row on the right set gets a more realistic looking hair shape by removing the small (incorrect) hair shape on top of his head. This effect may be due to the top-down global prior in the GLOC model, whereas simpler models such as the spatial CRF do not have this information. In addition, there were cases (such as the woman in the fifth row of the left set) where an additional face in close proximity to the centered face may confuse the model. In this case, the CRF and spatial CRF models make mistakes, but since the GLOC model has a strong shape model, it was able to find a more recognizable segmentation of the foreground face.

[0089] On the other hand, the GLOC model sometimes makes errors. Figure 6B shows typical failure examples. As seen, the model made significant errors in their hair regions. Specifically, in the first row, the hair of a nearby face is similar in color to the hair of the foreground face as well as the background, and the model incorrectly guesses more hair by emphasizing the hair shape prior, perhaps too strongly. In addition, there are cases in which occlusions cause problems, such as the third row. However, it is pointed out that the occlusions are frequently handled correctly by the model (e.g., the microphone in the third row of the left set in Figure 6A).

[0090] The model was also evaluated on the data set used by N. Wang, et al's in "What Are Good Parts For Hair Shape Modeling?", CVPR, 2012. This data set contains 1046 LFW (unfunneled) images whose pixels are manually labeled for 4 regions (Hair, Skin, Background, and Clothing). Following the evaluation setup, randomly split the data in half and use one half for training data and the other half for testing. The procedure is repeated five times, and report the average pixel accuracy as a final result.

[0091] First, the superpixels and features were generated for each image and then ran the GLOC model to get label guesses for each superpixel, and finally mapped back to pixels for evaluation. It was necessary to map to pixels at the end because the ground truth is provided in pixels. It was noted that even with a perfect superpixel labeling, this mapping already incurs approximately 3% labeling error. However, the approach was sufficient to obtain a good pixel-wise accuracy of 90.7% (91.7% superpixel-wise accuracy), which improves by 0.7% upon their best reported result of 90.0%. The ground truth for a superpixel is a normalized histogram of the pixel labels in the superpixel.

[0092] While the labeling accuracy is a direct way of measuring progress, an additional goal of this work is to build models that capture the natural statistical structure in faces. It is not an accident that human languages have words for beards, baldness, and other salient high-level attributes of human face appearance. These attributes represent coherent and repeated structure across the faces we see everyday. Furthermore, these attributes are powerful cues for recognition.

[0093] One of the most exciting aspects of RBMs and their deeper extensions are that these models can learn latent structure automatically. Recent work has shown that unsupervised learning models can learn meaningful structure without being explicitly trained to do so.

[0094] In the experiments, the GLOC model was run on all LFW images other than those used in training and validation, and sorted them based on each hidden unit activation. Each of the five columns in Figure 7 shows a set of retrieved images and their guessed labelings for a particular hidden unit. In many cases, the retrieved results for the hidden units form meaningful clusters. These units seem highly correlated with "lack of hair", "looking left", "looking right", "beard or occluded chin", and "big hair". Thus, the learned hidden units may be useful as attribute representations for faces.

[0095] Face segmentation and labeling is challenging due to the diversity of hair styles, head poses, clothing, occlusions, and other phenomena that are difficult to model, especially in a database like LFW. The proposed GLOC model combines the CRF and the RBM to model both local and global structure in face segmentations. This model has consistently reduced the error in face labeling over previous models which lack global shape priors. In addition, it is shown that the hidden units in the model can be interpreted as face attributes, which were learned without any attribute-level supervision.

[0096] In another aspect of this disclosure, modern low-level feature representations, such as SIFT and HOG, have had great success in visual recognition problems, yet there has been a growing body of work suggesting that the traditional approach of using only low-level features may be insufficient. Instead, significant performance gains can be achieved by introducing an intermediate set of features that capture higher-level semantic concepts beyond the plain visual cues that low-level features offer. One popular approach to introducing such mid-level features is to use semantic attributes. Specifically, each category can be represented by a set of semantic attributes, where some of these attributes can be shared by other categories. This facilitates the transfer of information between different categories and allows for improved generalization performance.

[0097] Typically, the attribute representation is obtained using the following process. First, a set of concepts is defined by the designer, and each instance in the training set has to be labeled with the presence or absence of each attribute. Subsequently, a classifier is trained for each of the attributes using the constructed training set. Furthermore, some additional feature selection schemes which utilize the attribute labels may be necessary in order to achieve satisfactory performance. Obtaining the semantic attribute representation is clearly a highly labor-intensive process. Furthermore, it is not clear how to choose the constituent semantic concepts for problems in which the shared semantic content is less intuitive (e.g., activity recognition in videos).

[0098] One approach to learning a semantic mid-level feature representation is based on latent Dirichlet allocation (LDA), which uses a set of topics to describe the semantic content. LDA has been very successful in text analysis and information retrieval, and has been applied to several computer vision problems. However, unlike linguistic words, visual words often do not carry much semantic interpretation beyond basic appearance cues. Therefore, the LDA has not been very successful in identifying mid-level feature representations.

[0099] Another line of work is the deep learning approach, such as deep belief networks (DBNs), which tries to learn a hierarchical set of features from unlabeled and labeled data. It has been shown that features in the upper levels of the hierarchy capture distinct semantic concepts, such as object parts. The DBNs can be effectively trained in a greedy layer-wise procedure using the restricted Boltzmann machine as a building block. The RBM is a bi-partite undirected graphical model that is capable of learning a dictionary of patterns from the unlabeled data. By expanding the RBM into a hierarchical representation, relevant semantic concepts can be revealed at the higher levels. RBMs and their extension to deeper architectures have been shown to achieve state-of-the-art results on image classification tasks.

[00100] In this aspect, it is proposed to combine the powers of topic models and DBNs into a single framework. That is, to learn mid-level features using the *replicated softmax* RBM (RS-RBM), which is an undirected topic model applied to bag-of-words data. Unlike other topic models, such as LDA, the RS-

RBM can be expanded into a DBN hierarchy by stacking additional RBM layers with binary inputs on-top of the first RS-RBM layer. Therefore, it is expected that features in higher levels can capture important semantic concepts that could not be captured by standard topic models with only a single layer (e.g., LDA).

[00101] As another contribution, a new approach is proposed to include class labels in training an RBM-like model. Although unsupervised learning can be effective in learning useful features, there is a lot to be gained by allowing some degree of supervision. To this end, a new extension of the RBM is developed which promotes a class-dependent use of dictionary elements. This can be viewed as a form of multi-task learning, and as such tends to improve the generalization performance.

[00102] The idea underlying this approach is to define an undirected graphical model using a factor graph with two kinds of factors: the first is an RBM-like type, and the second is related to a Beta-Bernoulli process (BBP) prior. The BBP is a Bayesian prior that is closely related to the Indian buffet process, and it defines a prior for binary vectors where each coordinate can be viewed as a feature for describing the data. The BBP has been used to allow for multi-task learning under a Bayesian formulation of sparse coding. This approach, which is referred to herein as the *Beta-Bernoulli Process Restricted Boltzmann Machine (BBP-RBM),* permits an efficient inference scheme using Gibbs sampling, akin to the inference in the RBM. Parameter estimation can also be effectively performed using Contrastive Divergence. Experimental results on object recognition show that the proposed model outperforms other baseline methods, such as LDA, RBMs, and previous state-of-the-art methods using attribute labels.

[00103] Background is first provided on RBMs and the BBP. The RBM defines a joint probability distribution over a hidden layer $h = [h_1, \ldots, h_K]^T$, where $h_k \in \{0,1\}$, and a visible layer $v = [v_1, \ldots, v_N]^T$, where $v_i \in \{0,1\}$. The joint probability distribution can be written as:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})). \qquad (1)$$

Here, the energy function of v, h is defined as

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^T \mathbf{W} \mathbf{v} - \mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}, \qquad (2)$$

where $\mathbf{W} \in \mathbb{R}^{K \times N}, \mathbf{b} \in \mathbb{R}^{K}, \mathbf{c} \in \mathbb{R}^{N}$ are parameters.

[00104] It is straightforward to show that the conditional probability distributions take the form

$$p(h_k = 1|\mathbf{v}) = \sigma\left(\sum_i w_{k,i} v_i + b_k\right), \quad (3)$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(\sum_k w_{k,i} h_k + c_i\right), \quad (4)$$

where $\sigma(x) =)1 + e^{-x})^{-1}$ is the sigmoid function. Inference can be performed using Gibbs sampling, alternating between sampling the hidden and visible layers. Although computing the gradient of the log-likelihood of training data is intractable, the Contrastive Divergence approximation can be used to approximate the gradient. Other types of training method are also contemplated by this disclosure.

[00105] The RBM can be extended to the case where the observations are word counts in a document. The word counts are transformed into a vector of binary digits, where the number of 1's for each word in the document equals its word count. A single hidden layer of a binary RBM then connects to each of these binary observation vectors (with weight sharing), which allows for modeling of the word counts. The model can be further simplified such that it deals with the word count observations directly, rather than with the intermediate binary vectors. Specifically, let N denote the number of words in the dictionary, and let v_i $(i = 1, \ldots, N)$ denote the number of times word i appears in the document, then the joint probability distributions of the binary hidden layer **h** and the observed word counts **v** is of the same form as in Equations (1) and (2), where the energy of **v, h** is defined as

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^T \mathbf{W} \mathbf{v} - D\mathbf{b}^T \mathbf{h} - \mathbf{c}^T \mathbf{v}, \quad (5)$$

and $D = \sum_{i=1}^{N} v_i$ is the total word count in a document.

[00106] Inference is performed using Gibbs sampling, where the posterior for the hidden layer takes the form

$$p(h_k = 1|\mathbf{v}) = \sigma\left(\sum_i w_{k,i} v_i + D b_k\right). \quad (6)$$

Sampling from the posterior of the visible layer is performed by sampling D times from the following multinominal distribution:

$$p_i = \frac{\exp(\sum_{k=1}^{K} h_k w_{k,i} + c_i)}{\sum_{i=1}^{N} \exp(\sum_{k=1}^{K} h_k w_{k,i} + c_i)}, i = 1, \ldots, N, \quad (7)$$

and setting v_i to the number of times the index i appears in the D samples. Parameter estimation is performed in the same manner as the case of the RBM with binary observations.

[00107] BBP is a Bayesian generative model for binary vectors, where each coordinate can be viewed as a feature for describing the data. In this work, a finite approximation to the BBP is used which can be described using the following generative model. Let $f_1, \ldots, f_K \in \{0,1\}$ denote the elements of a binary vector, then the BBP generates f_k according to:

$$\begin{aligned} \pi_k &\sim \text{Beta}\left(\frac{\alpha}{K}, \beta(K-1)/K\right) \\ f_{k \sim} &\text{Bernoulli}\,(\pi_k), \end{aligned} \quad (8)$$

where α, β are positive constants (hyperparameters), and the notation $\pi = [\pi_1, \ldots, \pi_K]^T$. Equation (8) implies that if π_k is close to 1 then f_k is more likely to be 1, and vice versa. Since the Beta and Bernoulli distributions are conjugate, the posterior distribution for π_k also follows a Beta distribution. In addition, for a sufficiently large K and reasonable choices of α and β, most π_k will be close to zero, which implies a sparsity constraint on f_k.

[00108] Furthermore, by drawing a different π_k for each class, a unique class-specific sparsity structure can be imposed, and such a prior allows for multi-task learning. The BBP has been used to allow for multi-task learning under a Bayesian formulation of sparse coding. The multi-task paradigm promotes sharing of information between related groups, and therefore can lead to improved generalization performance. Motivated by this observation, an extension of the RBM is proposed that incorporates a BBP-like factor and extend to a deeper architecture.

[00109] Figure 8 illustrates an exemplary mid-level feature extraction scheme. A low-level feature extraction method is used, where the image is first partitioned into a 3×2 grid, and HOG, texture, and color features are extracted from each of the cells, as well as from the entire image. In order to obtain the

bag-of-words representation, first compute the histogram over the visual words, and then obtain the word counts by multiplying each histogram with a constant (e.g., the constant 200 is used throughout this work) and rounding the numbers to the nearest integer values.

[00110] The word counts are used as the inputs to RS-RBMs (or BBP-RS-RBMs which will be described below), where different RS-RBM units are used for each of the histograms. The binary outputs of all the RS-RBM units are concatenated and fed into a binary RPB (or a binary BBP-RBM) at the second layer. The outputs of the hidden units of the second layer are then used as input to the third layer binary RPB, and similarly to any higher layers. Training the DBN is performed in a greedy layer-wise fashion, starting with the first layer and proceeding in the upward direction.

[00111] Each of the RS-RBM units independently captures important patterns which are observed within its defined feature type and spatial extent. The binary RBM in the second layer captures higher-order dependencies between the different histograms in the first layer. The binary RBMs in higher levels could model further high-order dependencies, which hypothesize to be related to some semantic concepts. Below find associations between the learned features and manually specified semantic attributes.

[00112] The feature vector which is used for classification is obtained by concatenating the outputs of all the hidden units from all the layers of the learned DBN. Given a training set, compute the feature vector for every instance and train a multi-class classifier. Similarly, for every previously unseen test instance, compute its feature vector and classify it using the trained classifier.

[00113] Next, the BBP-RBM is developed, both when assuming that all the training examples are unlabeled, and also when each example belongs to one of C classes. These two versions are referred to as single-task BBP-RBM and multi-task BBP-RBM, respectively. The single-task version can be considered as a new approach to introduce sparsity into the RBM formulation, which is an alternative to the common approach of promoting sparsity through regularization. It is also related to “dropout”, which randomly sets individual hidden units to zeros during training and has been reported to reduce overfitting when training deep convolutional neural networks. The BBP-RBM uses a factor

graph formulation to combine two different types of factors: the first factor is related to the RBM, and the second factor is related to the BBP. Combining these factors together leads to an undirected graphical model for which we develop efficient inference and parameter estimation schemes.

[00114] A binary selection vector is defined $\mathrm{f} = [f_1, \ldots, f_K]^T$ that is used to choose which of the K hidden units to activate. This approach is to define an undirected graphical model in the form of a factor graph with two types of factors, as shown in Figure 9A for the single-task case and Figure 9B for the multi-task cases. The first factor is obtained as an unnormalized RBM-like probability distribution which includes the binary selection variables **f**:

$$g_a(\mathrm{v}, \mathrm{h}, \mathrm{f}) = exp\big(-E(\mathrm{v}, \mathrm{h}, \mathrm{f})\big), \tag{9}$$

where the energy term takes the form

$$E(\mathrm{v}, \mathrm{h}, \mathrm{f}) = -(\mathrm{f} \odot \mathrm{h})^T \mathrm{W}\mathrm{v} - b^T(\mathrm{f} \odot \mathrm{h}) - c^T\mathrm{v}, \tag{10}$$

and $\odot$ denotes element-wise vector multiplication.

[00115] The second factor is obtained from the BBP generative model (described in Equation (8)) as follows:

$$g_b\left(\{\mathrm{f}^{(j)}\}_{j=1}^{M}, \pi\right) = \prod_{k=1}^{K} \pi_k^{\sum_{j=1}^{M} f_k^{(j)}} (1-\pi_k) \sum_{j=1}^{M} \left(1 - f_k^{(j)}\right) \times \pi_k^{\alpha/K-1} (1-\pi_k)^{\beta(K-1)/K-1} \tag{11}$$

where j denotes the index of the training sample, and M denotes the number of training samples.

[00116] Using the factor graph description in Figure 9A, the probability distribution for the single-task BBP-RBM takes the form

$$p\left(\{\mathrm{v}^{(j)}, \mathrm{h}^{(j)}, \mathrm{f}^{(j)}\}_{j=1}^{M}, \pi\right) \propto g_b\left(\{\mathrm{f}^{(j)}\}_{j=1}^{M}, \pi\right) \times \prod_{j=1}^{M} g_{a(\mathrm{v}^{(j)}, \mathrm{h}^{(j)}, \mathrm{f}^{(j)})}. \tag{12}$$

Using the factor graph description in Figure 9B, the joint probability distribution for the multi-task case takes the form

$$p\left(\{\mathrm{v}^{(j)},\mathrm{h}^{(j)},\mathrm{f}^{(j)}\}_{j=1}^{M},\{\pi^{(c)}\}_{c=1}^{C}\right)$$
$$\propto \prod_{c=1}^{C}\left(g_b\left(\{\mathrm{f}^{(j_c)}\}_{j=1}^{M_c},\pi^{(c)}\right)\prod_{j_c=1}^{M_c} g_a\left(\mathrm{v}^{(j_c)},\mathrm{h}^{(j_c)},\mathrm{f}^{(j_c)}\right)\right), \quad (13)$$

where C denotes the number of different classes in the training set, and uses the notation j_c to denote the unique index of the training instance which belongs to class c, and M_c denotes the number of training instances which belong to class c.

[00117] Similarly to the RBM, inference in the BBP-RBM can be performed using Gibbs sampling. The posterior probability distributions are provided only for the multi-task case, since the single-task can be obtained as a special case by setting $C = 1$. Sampling from the joint posterior probability distribution of **h** and **f** can be performed using

$$p\left(h_k^{(j_c)} = x, f_k^{(j_c)} = y \mid -\right) \propto \begin{cases} \pi_k^{(c)} e_k^{\delta^{(j_c)c}}, & x=1, y=1 \\ \pi_k^{(c)}, & x=0, y=1 \\ 1-\pi_k^{(c)}, & x=0, y=0 \\ 1-\pi_k^{(c)}, & x=1, y=0 \end{cases} \quad (14)$$

where " $-$ " denotes all other variables, and define $\delta_k^{(j_c)} = \sum_i w_{k,i}\, v_i^{(j_c)} + b_k$ for binary inputs, or $\delta_k^{(j_c)} = \sum_i w_{k,i} v_i^{(j_c)} + D b_k$ for word count observations.

[00118] The posterior probability for $\pi^{(c)}$ takes the form

$$p\left(\pi_k^{(c)} \mid -\right) =$$
$$\mathrm{Beta}\left(\frac{\alpha}{K} + \sum_{j_c=1}^{M_c} f_k^{(j_c)},\ \beta(K-1)/K + \sum_{j_c=1}^{M_c}\left(1 - f_k^{(j_c)}\right)\right) \quad (15)$$

Sampling from the posterior of the visible layer is performed in a similar way that was discussed above for the RBM with either binary or word count observations, where the only difference is that **h** is replaced by $\mathrm{f} \odot \mathrm{h}$.

[00119] From Equation (14), it is observed that if $\pi_k^{(j_c)} = 1$ then the BBP-RBM reduces to the standard RBM, since the posterior probability distribution for $h_k^{(j_c)}$ becomes $p\left(h_k^{(j_c)} = 1 \middle| -\right) = \delta\left(\delta_k^{(j_c)}\right)$ (i.e., the standard RBM has the same posterior probability for $h_k^{(j_c)}$).

[00120] Using the property of conditional expectation, it can be shown that the gradient of the log-likelihood of $\mathrm{v}^{(j_c)}$ with respect to the parameter $\theta \in \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ takes the form

$$\frac{\partial \log p(\mathrm{v}^{(j_c)})}{\partial \theta} = \mathbb{E}_{\pi^{(c)}}\left[-\mathbb{E}_{\mathrm{h,\,f,}|\pi^{(c)},\mathrm{v}^{(j_c)}}\left[\frac{\partial}{\partial \theta}E\left(\mathrm{v}^{(j_c)},\mathrm{h,\,f}\right)\right] + \mathbb{E}_{\mathrm{h,\,f,\,v}|\pi^{(c)}}\left[\frac{\partial}{\partial \theta}E(\mathrm{v,\,h,f})\right]\right]. \quad (16)$$

The expression cannot be evaluated analytically; however, it is noted that the first inner expectation does admit an analytical expression, whereas the second inner expectation is intractable. It is proposed to use an approach similar to Contrastive Divergence to approximate Equation (16). First, sample $\pi^{(c)}$ using Gibbs sampling, and then use a Markov chain Monte-Carlo approach to approximate the second inner expectation. The batch version of this approach is summarized in Algorithm 1 below. In practice, an online version is used where the parameters are updated incrementally using mini-batches. The parameters $\{\pi^{(c)}\}_{c=1}^{C}$ are resampled only after a full sweep over the training set is finished.

Algorithm 1 Batch Contrastive Divergence training for the multi-task BBP-RBM

Input: Previous samples of $\{\pi^{(c)}\}_{c=1}^{(c)}$, training samples $\{\mathrm{v}^{(j)}\}_{j=1}^{M}$, and learning rate λ

- For $c = 1, \ldots, C$, sample $\pi_{new}^{(c)}$ as follows
  1. Sample $\mathrm{h}^{(j_c)}, \mathrm{f}^{(j_c)}|\pi^{(c)}, \mathrm{v}^{(j_c)}, \forall\, j_c = 1, \ldots, M_c$ using Equation (14).
  2. Sample $\pi_{new}^{(c)}$using Equation (15).
- For $c = 1, \ldots, C, j_c = 1, \ldots, M_c$
  1. Sample $\mathrm{h}^{(j_c,0)}, \mathrm{f}^{(j_c,0)}|\pi_{new}^{(c)}, \mathrm{v}^{(j_c)}$.
  2. Sample $\mathrm{v}^{(j_c,1)}|\pi_{new}^{(c)}, \mathrm{h}^{(j_c,0)}, \mathrm{f}^{(j_c,0)}$.
  3. Sample $\mathrm{h}^{(j_c,1)}, \mathrm{f}^{(j_c,1)}|\pi_{new}^{(c)}, \mathrm{v}^{(j_c,1)}$.
- Update each of the parameters $\theta \in \{\mathrm{W, b, c}\}$ using

$$\theta \leftarrow \theta - \lambda \sum_{c=1}^{C}\sum_{j_c=1}^{M_c}\left(\frac{\partial}{\partial \theta}E\left(\mathrm{v}^{(j_c)}, \mathrm{h}^{(j_c,0)}, \mathrm{f}^{(j_c,0)}\right) - \frac{\partial}{\partial \theta}E\left(\mathrm{v}^{(j_c,1)}, \mathrm{h}^{(j_c,1)}, \mathrm{f}^{(j_c,1)}\right)\right)$$

[00121] When using the BBP-RBM in the DBN architecture described in Figure 8 there is an added complication of dealing with the variable π since it cannot be marginalized efficiently. One solution is to train each layer of a BBP-RBM as described in the previous section. However, when computing the output of the hidden units to be fed into the consecutive layer, choose $\pi_k^{(c)} = 1, \forall c = 1, \ldots, C, k = 1, \ldots, K$, which corresponds to the output of a standard RBM (as

explained above). Using this approach, avoid the issues which would otherwise arise during the recognition stage (i.e., class labels are unknown for test examples).

[00122] The features learned by the BBP-RBM were evaluated using two datasets that were developed by A. Farhadi, et al's in "Describing Objects By Their Attributes, CVPR, 2009, which include annotation for labeled attributes. The two datasets are referred to as the PASCAL and Yahoo datasets. Object classification experiments are performed within the PASCAL dataset and also across the two datasets (i.e., learning the BBP-RBM features using the PASCAL training set, and performing classification on the Yahoo dataset). Finally, the semantic content of the features are examined by finding correspondences between the learned features and the manually labeled attributes available for the PASCAL dataset. These correspondences were also used to perform attribute localization experiments, by predicting the bounding boxes for several of the learned mid-level features.

[00123] The PASCAL dataset is comprised of instances corresponding to 20 different categories, with pre-existing splits into training and testing sets, each containing over 6400 images. The categories are: person, bird, cat, cow, dog, horse, sheep, airplane, bicycle, boat, bus, car, motorcycle, train, bottle, chair, dining-table, potted-plant, sofa and tv/monitor. The Yahoo dataset contains 2644 images with 12 categories which are not included in the PASCAL dataset. Additionally, there are annotations for 64 attributes which are available for all the instances in the PASCAL and Yahoo datasets. The feature types used are: 1000 dimensional HOG histogram, 128 dimensional color histogram, and 256 dimensional texture histogram. An eight dimensional edge histogram was used as well; however, it was not used in the RBM and BBP-RBM based experiments since the code to extract the edge features and the corresponding descriptors were not available online. Note that not using the edge features in the methods may give an unfair disadvantage when comparing results that used all the base features. The HOG, color, and texture descriptors which were used are identical to those in the PASCAL and Yahoo datasets. When learning an RBM based model, 800 hidden units were used for the HOG histogram, 200 hidden units for the color histogram, and 300 units for the texture

histogram. The number of hidden units for the upper layers was 4000 for the second layer, and 2000 for the third layer.

[00124] In Table 5, the test classification accuracy is compared for the PASCAL dataset using features that were learned with the following methods: LDA, the standard RBM, the RBM with sparsity regularization (sparse RBM), the single-task BBP-RBM, and the multi-task BBP-RBM. The LDA features were the topic proportions learned for each of the histograms, and 50 topics were used for each histogram. For evaluating features, the multi-class linear SVM was used in all the experiments. When performing cross validation, the training set was partitioned into two sets. The first was used to learn the BBP-RBM features, and the second was used as a validation set. For both the overall classification accuracy and the mean per-class classification accuracy, the sparse RBM outperformed the standard RBM and LDA, but it performed slightly worse than the single-task BBP-RBM. This could suggest that the single-task BBP-RBM is an alternative approach to inducing sparsity in the RBM. Furthermore, the multi-task BBP-RBM outperformed all other methods, particularly for the mean per-class classification rate. Adding more layers generally improved the classification performance; however, the improvement reached saturation at approximately 2-3 layers.

Table 5

	Overall			Mean per-class		
# Layers	1	2	3	1	2	3
LDA	54.0	-	-	32.1	-	-
RBM	55.5	55.9	56.5	62.6	33.6	33.8
sparse-RBM	60.0	60.8	61.0	40.5	41.1	41.8
single-task BBP-RBM	61.7	62.0	61.7	42.3	42.3	41.7
multi-task BBP-RBM	62.5	63.2	63.2	42.7	45.5	46.1

[00125] In Table 6, the classification results obtained using the multi-task BBP-RBM are compared to the results reported in A. Farhadi, et al's "Describing Objects By Their Attributes", CVPR, 2009 and Y. Wang, et al's "A Discriminative Latent Model Of Object Classes And Attributes", ECCV, 2010 for the same task. Note that the baseline methods were adapted to exploit the information from the labeled attributes (which the BBP-RBM did not use). Note that attribute annotations are very expensive to obtain, and for many visual

recognition problems, such as activity recognition in videos, it is even harder to identify and label the semantic content that is shared by different types of classes. The results show that, even though the method did not use the attribute annotation, it significantly improved both the overall classification accuracy and the mean per-class accuracy in comparison to the baseline methods.

Table 6

	Using attributes		Without attributes	
Method	Overall	Per-class	Overall	Per-class
[7]	59.4	37.7	58.5	34.3
[29] w/loss-1	62.16	46.25	58.77	38.52
[29] w/loss-2	59.15	50.84	53.74	44.04
BBP-RPM	-	-	63.2	46.1

[00126] An important aspect of evaluating the features in the degree to which they generalize well across different datasets. To this end, the PASCAL training set was used to learn the features and evaluated their performance on the Yahoo dataset into different proportions of training samples and compared the performance when using the multi-task BBP-RBM and base features, respectively. Table 7 summarizes the test accuracy averaged over 10 random trials for several training set sizes. The results suggest that the method using the BBP-RBM features can recognize new categories from the Yahoo dataset with fewer training samples, as compared to using the base features. For example, the overall classification performance with the BBP-RBM features using only 20% of the dataset for training is comparable to or better than that with the base features using 60% of the dataset for training.

Table 7

	Base features		BBP-RBM features	
Training %	Overall	Per-class	Overall	Per-class
10%	56.3	44.2	65.8	54.1
20%	61.0	48.2	70.6	60.0
30%	64.5	49.9	73.6	62.4
40%	68.3	51.4	75.3	64.8
50%	71.0	52.2	77.1	66.6
60%	71.0	52.5	78.4	68.4

[00127] In this experiment, the degree to which the features learned using the BBP-RBM demonstrate identifiable semantic concepts was evaluated. For each feature and labeled attribute pair, the score given by Equation (3) was used to predict the presence of manually labeled semantic attributes in each training example and computed the area under the ROC curve over the PASCAL training data. The feature corresponding to each attribute is determined as that which has the largest area under the ROC curve. Figure 10 shows the corresponding area under the ROC curve for every attribute on the PASCAL test data (i.e., using the training set to determine the correspondences, and the test set to compute the ROC area). The area under the ROC curve obtained using attribute classifiers (linear SVMs trained using the attribute labels and the base features) is also shown together. The figure shows that the learned features without using attribute labels performed reasonably well, and some learned features performed comparably to the attribute classifiers that were trained using the attribute labels. It is noted that all the semantic attributes were associated to features in either the second layer or the third layer in Figure 8, which supports the hypothesis that the higher levels of the DBN can capture semantic concepts.

[00128] Experiments were also performed where the mid-level features corresponding to the attributes “snout”, “skin”, and “furry” were used to predict the bounding boxes of these attributes. For fine-grained localization, simple sliding-window detection was run using bounding boxes of different aspect ratios on each image, and show only the first few non-overlapping windows that achieved the best scores. Although there were some misdetections in the “skin” case, appropriate bounding boxes were identified. Note that there were no bounding boxes available for these attributes in the training set (i.e., the bounding boxes were provided only for the entire objects); yet in some cases the BBP-RBM could localize the subparts of the categories which the attributes describe.

[00129] In the experiments, the hyperparameter values $\alpha = 1$, $\beta = 5$ were used for the BBP-RBM. It was observed that the exact choice of these hyperparameter had very little effect on the performance. The parameters **W, b** and **c** were initialized by drawing from a zero-mean isotropic Gaussian with standard deviation 0.001. ℓ_2 regularization was added for the elements of **W**, and used the regularization hyperparameter 0.001 for the first layer and 0.01 for

the second and third layers. A target sparsity of 0.2 was used for the sparse RBM. These hyperparameters were determined by cross validation.

[00130] In sum, the BBP-RBM is proposed as a new method to learn mid-level feature representations. The BBP-RBM is based on a factor graph representation that combines the properties of the RBM and the Beta-Bernoulli process. The method can induce category-dependent sharing of learned features, which can be helpful in improving the generalization performance.

[00131] One of the most challenging aspects of image recognition is the large amount of intra-class variability, due to factors such as lighting, background, pose, and perspective transformation. For tasks involving a specific object category, such as face verification, this intra-class variability can often be much larger than inter-class differences. This variability can be seen in sample images from Labeled Faces in the Wild (LFW), a data set used for benchmarking unconstrained face verification performance. The task in LFW is, given a pair of face images, determine if both faces are of the same person (matched pair), or if each shows a different person (mismatched pair).

[00132] Recognition performance can be significantly improved by removing undesired intra-class variability, by first aligning the images to some canonical pose or configuration. For instance, face verification accuracy can be dramatically increased through image alignment, by detecting facial feature points on the image and then warping these points to a canonical configuration. This alignment process can lead to significant gains in recognition accuracy on real-world face verification, even for algorithms that were explicitly designed to be robust to some misalignment. Therefore, the majority of face recognition systems evaluated on LFW currently make use of a preprocessed version of the data set known as LFW-a (http://www.openu.ac.il/home/hassner/data/lfwa/), where the images have been aligned by a commercial fiducial point-based supervised alignment method. Fiducial point (or landmark-based) alignment algorithms, however, require a large amount of supervision or manual effort. One must decide which fiducial points to use for the specific object class, and then obtain many example image patches of these points. These methods are thus hard to apply to new object classes, since all of this manual collection of data must be re-

done, and the alignment results may be sensitive to the choice of fiducial points and quality of training examples.

[00133] An alternative to this supervised approach is to take a set of poorly aligned images (e.g., images drawn from approximately the same distribution as the inputs to the recognition system) and attempt to make the images more similar to each other, using some measure of joint similarity such as entropy. This framework of iteratively transforming images to reduce the entropy of the set is known as congealing, and was originally applied to specific types of images such as binary handwritten characters and magnetic resonance image volumes. Congealing was extended to work on complex, real-world object classes such as faces and cars. However, this required a careful selection of hand-crafted feature representation (SIFT) and soft clustering, and does not achieve as large of an improvement in verification accuracy as supervised alignment (LFW-a).

[00134] In accordance with yet another aspect of this disclosure, a novel combination of unsupervised alignment and unsupervised feature learning is proposed, specifically by incorporating deep learning into the congealing framework. Through deep learning, one can obtain a feature representation tuned to the statistics of the specific object class desired to be aligned, and capture the data at multiple scales by using multiple layers of a deep learning architecture. Further, a group sparsity constraint can be incorporated into the deep learning algorithm, leading to a topographic organization on the learned filters, and show that this leads to improved alignment results.

[00135] The congealing framework is first reviewed. Two terms used in congealing are the distribution field (DF) and the location stack. Let $X = \{1,2,\ldots,M\}$ be the set of all feature values. For example, letting the feature space be intensity values, $M = 2$ for binary images and $M = 256$ for 8-bit grayscale images. A distribution field is a distribution over χ at each location in the image representation; *e.g.,* for binary images, a DF would be a distribution over $\{0,1\}$ at each pixel in the image. One can view the DF as a generative independent pixel model of images, by placing a random variable X_i at each pixel location i. An image then consists of a draw from the alphabet χ for each Xi according to the distribution over χ at the ith pixel of the DF. Given a set of images, the location

stack is defined as the set of values, with domain χ, at a specific location across a set of images. Thus, the empirical distribution at a given location of a DF is determined by the corresponding location stack.

[00136] Congealing proceeds by iteratively computing the empirical distribution defined by a set of images, then for each image, choosing a transformation (e.g., the set of similarity transformations) that reduces the entropy of the distribution field. Figure 11 illustrates congealing on one dimensional binary images. Under an independent pixel model and uniform distribution over transformations, minimizing the entropy of the distribution field is equivalent to maximizing the likelihood according to the distribution field.

[00137] Once congealing has been performed on a set of images *(e.g.,* a training set*), funneling* can be used to quickly align additional images, such as from a new test set. This is done by maintaining the sequence of DFs from each iteration of congealing. A new image is then aligned by transforming it iteratively according to the sequence of saved DFs, thereby approximating the results of congealing on the original set of images as well as the new test image. As mentioned earlier, congealing was extended to work on complex object classes, such as faces, by using soft clustering of SIFT descriptors as the feature representation. This congealing algorithm will be referred to as SIFT congealing; whereas, the proposed extension is referred to *deep congealing.*

[00138] To incorporate deep learning within congealing, the convolutional restricted Boltzmann machine (CRBM) is used in conjunction with convolutional deep belief network (CDBN). The CRBM is an extension of the restricted Boltzmann machine, which is a Markov random field with a hidden layer and a visible layer (corresponding to image pixels in computer vision problems), where the connection between layers is bipartite. In the CRBM, rather than fully connecting the hidden layer and visible layer, the weights between the hidden units and the visible units are local *(i.e.,* 10 x 10 pixels instead of full image) and shared among all hidden units. An illustration of CRBM can be found in Figure 12. The CRBM has three sets of parameters: (1) K convolution filter weights between the hidden nodes and the visible nodes, where each filter is N_W x N_W pixels *(i.e.,* $W^k \ \mathbb{R}^{N_W x N_W}$, $k = 1, \ldots, K$*);* (2) hidden biases $b^k \epsilon \mathbb{R}$ that are shared

among hidden nodes; and (3) visible bias $c \in \mathbb{R}$ that is shared among visible nodes.

[00139] To make CRBMs more scalable, probabilistic max-pooliing was developed by Lee et al's in "Unsupervised Learning Of Hierarchical Representations With Convolutional Deep Belief Networks", Communications of the ACM 54 (10): 95-103, 2011. Probabilistic max-pooling is a technique for incorporating local translation invariance. Max-pooling refers to operations where a local neighborhood *(e.g.,* 2 x 2 grid) of feature detection outputs is shrunk to a pooling node by computing the maximum of the local neighbors. Max-pooling makes the feature representation more invariant to local translations in the input data, and has been shown to be useful in computer vision.

[00140] Letting $P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(-\mathrm{E}(\mathbf{v}, \mathbf{h}))$, the energy function of the probabilistic max- pooling CRBM is defined as follows:

$$E(\mathrm{v},\mathrm{h}) = \sum_{k=1}^{K}\sum_{i,j=1}^{N_H} h_{ij}^k\left(\widetilde{W}^k * \mathrm{v}\right)_{ij} + \sum_{r,s=1}^{N_v}\frac{1}{2}v_{rs}^2 - \sum_{k=1}^{K} b^k \sum_{i,j=1}^{N_H} h_{ij}^k - c\sum_{r,s=1}^{N_v} v_{rs}$$

$$\text{s.t.} \quad \sum_{(i,j)\in B_\alpha} h_{ij}^k \leq 1, \forall k, \alpha$$

Here, $\widetilde{W}^k$ refers to flipping the original filter W^k in both upside-down and left-right directions, and $*$ denotes convolution. B_α refers to a C x C block of locally neighboring hidden units *(i.e.,* pooling region) $h_{i,j}^k$ that are pooled to a pooling node p_α^k. Real-valued visible units are used in the first-layer CRBM; however, binary-valued visible units are used when constructing the second-layer CRBM. The CRBM can be trained by approximately maximizing the log-likelihood of the unlabeled data via contrastive divergence.

[00141] After training a CRBM, it can be used to compute the posterior of the pooling units given the input data. These pooling unit activations can be used as input to further train the next layer CRBM. By stacking the CRBMs, the algorithm can capture high-level features, such as hierarchical object-part decompositions. After constructing a convolutional deep belief network, (approximate) inference of the whole network is performed in a feedforward (bottom-up) manner. Specifically, letting $I\left(h_{ij}^k\right) \triangleq b^k + \left(\widetilde{W}^k * \mathrm{v}\right)_{ij}$, the pooling unit activations can be inferred as a softmax function:

$$P(p_\alpha^k = 1|\mathrm{v}) = \frac{\sum_{(i',j') \in B_\alpha} \exp\left(I\left(h_{i'j'}^k\right)\right)}{1 + \sum_{i',j'} \in B_\alpha \exp\left(I\left(h_{i'j'}^k\right)\right)}$$

Given a set of poorly aligned face images, the goal is to iteratively transform each image to reduce the total entropy over the pooling layer outputs of a CDBN applied to each of the images. For a CDBN with *K* pooling layer groups, there are *K* location stacks at each image location (after max-pooling), over a binary distribution for each location stack. Given *N* unaligned face images, let *P* be the number of pooling units in each group in the top-most layer of the CDBN. Using the pooling unit probabilities, with the interpretation that the pooling unit can be considered as a mixture of sub-units that are on and off. Letting $p_\alpha^{k,(n)}$ be the pooling unit α in group *k* for image *n* under some transformation T^n, define $D_\alpha^k(1) = \frac{1}{N}\sum_{n=1}^N p_\alpha^{k,(n)}$ and $D_\alpha^k(0) = 1 - D_\alpha^k(1)$. Then, the entropy for a specific pooling unit is $H(D_\alpha^k) = -\sum_{s\epsilon\{0,1\}} D_\alpha^k(s)\log(D_\alpha^k(s))$. At each iteration of congealing, find a transformation for each image that decreases the total entropy $\sum_{k=1}^K \sum_{\alpha=1}^P H(D_\alpha^k)$. Note that if $K = 1$, this reduces to the traditional congealing formulation on the binary output of the single pooling layer.

[00142] As congealing reduces entropy by performing local hill-climbing in the transformation parameters, a key factor in the success of congealing is the smoothness of this optimization landscape. In SIFT congealing, smoothness is achieved through soft clustering and the properties of the SIFT descriptor. Specifically, to compute the descriptor, the gradient is computed at each pixel location and added to a weighted histogram over a fixed number of angles. The histogram bins have a natural circular topology. Therefore, the gradient at each location contributes to two neighboring histogram bins, weighted using linear interpolation. This leads to a smoother optimization landscape when congealing. For instance, if a face is rotated a fraction of the correct angle to put it into a good alignment, there will be a corresponding partial decrease in entropy due to this interpolated weighting.

[00143] In contrast, there is no topology on the filters produced using standard learning of a CRBM. This may lead to plateaus or local minima in the optimization landscape with congealing, for instance, if one filter is a small rotation of another filter, and a rotation of the image causes a section of the face

to be between these two filters. This problem may be particularly severe for filters learned at deeper layers of a CDBN. For instance, a second-layer CDBN trained on face images would likely learn multiple filters that resemble eye detectors, capturing slightly different types and scales of eyes. If these filters are activating independently, then the resulting entropy of a set of images may not decrease even if eyes in different images are brought into closer alignment.

[00144] A CRBM is generally trained with sparsity regularization, such that each filter responds to a sparse set of input stimuli. A smooth optimization for congealing requires that, as an image patch is transformed from one such sparse set to another, the change in pooling unit activations is also gradual rather than abrupt. Therefore, it would be beneficial to learn filters with a linear topological ordering, such that when a particular pooling unit p_α^k at location α and associated with filter *k* is activated, the pooling units at the same location, associated with nearby filters, *i.e.,* $p_\alpha^{k'}$ for *k'* close to *k,* will also have partial activation. To learn a topology on the learned filters, add the following group sparsity penalty to the learning objective function *(i.e.,* negative log-likelihood): $\mathcal{L}_{sparsity} = \lambda \sum_{k,\alpha} \sqrt{\sum_{k'} \omega_{k'-k} \left(p_\alpha^k\right)^2}$, where ω_d is a Gaussian weighting, $\omega_d \propto \exp\left(-\frac{d^2}{2\sigma^2}\right)$. Let the term *array* be used to refer to the set of pooling units associated with a particular filter, *i.e.,* p_α^k for all locations α. This regularization penalty is a sum (L^1 norm) of L^2 norms, each of which is a Gaussian weighting, centered at a particular array, of the pooling units across each array at a specific location. In practice, rather than weighting every array in each summand, use a fixed kernel covering five consecutive filters, *i.e.,* ω_d 0 for $|d| > 2$.

[00145] The rationale behind such a regularization term is that, unlike an L^2 norm, an L^1 norm encourages sparsity. This sum of L^2 norms thus encourages sparsity at the group level, where a group is a set of Gaussian weighted activations centered at a particular array. Therefore, if two filters are similar and tend to both activate for the same visible data, a smaller penalty will be incurred if these filters are nearby in the topological ordering, as this will lead to a more sparse representation at the group L^2 level. To account for this penalty term, augment the learning algorithm by taking a step in the negative derivative

with respect to the CRBM weights. Define $\alpha(i,j)$ as the pooling location associated with position (i,j), and J as $j_{ij}^{k,k'} = \frac{1}{\sqrt{\sum_{k''} \omega_{k''-k'} \left(p_{\alpha(i,j)}^{k''}\right)^2}} p_{\alpha(i,j)}^{k}\left(1 - p_{\alpha(i,j)}^{k}\right)h_{ij}^{k}$. It can be written the full gradient as $\nabla_{W^k}\mathcal{L}_{sparsity} = \lambda \sum_{k'} \omega_{k-k'}(v * \tilde{J}^{k,k'})$, where * denotes convolution and $\tilde{J}^{k,k'}$ means $J^{k,k'}$ flipped horizontally and vertically. Thus, the gradient can be efficiently computed as a sum of convolutions.

[00146] Following the procedure given by Sohn *et al.* in "Efficient Learning Of Sparse, Distributed, Convoluational Feature Representations For Object Recognition", ICCVm 2011, initialize the filters using expectation-maximization under a mixture of Gaussians/Bernoullis, before proceeding with CRBM learning. Therefore, when learning with the group sparsity penalty, periodically reorder the filters using the following greedy strategy. Taking the first filter, iteratively add filters one by one to the end of the filter set, picking the filter that minimizes the group sparsity penalty.

[00147] Three different convolutional DBN models are used as the feature representation for deep congealing. First, a one-layer CRBM is learned from the Kyoto images (see, http://www.cnbc.emu.edu/cplab/data_kyoto.html), a standard natural image data set, to evaluate the performance of congealing with self-taught CRBM features. Next, a one-layer CRBM is learned from LFW face images, to compare performance when learning the features directly on images of the object class to be aligned. Finally, a two-layer CDBN is learned from LFW face images, to evaluate performance using higher-order features. For all three models, compare learning the weights using the standard sparse CDBN learning, as well as learning with group sparsity regularization. Visualizations of the top layer weights of the two-layer CDBN are given in Figure 13, demonstrating the effect of adding the sparsity regularization term.

[00148] K = 32 filters are used for the one-layer models and K = 96 are used in the top layer of the two-layer models. During learning, a pooling size of 5x5 is used for the one-layer models and 3x3 in both layers of the two-layer model. $\sigma^2 = 1$ is used in the Gaussian weighting for group sparsity regularization. For computing the pooling layer representation to use in congealing, modify the pooling size to 3x3 for the one-layer models and 2x2 for the second layer in the

two-layer model, and adjusted the hidden biases to give an expected activation of 0.025 for the hidden units. In Figure 14, a selection of images is shown under several alignment methods. Each image is shown in its original form, and aligned using SIFT Congealing, Deep Congealing with topology, using a one-layer and two-layer CDBN trained on faces, and the LFW-a alignment.

[00149] The effect of alignment on verification accuracy is evaluated using View 1 of LFW. For the congealing methods, 400 images from the training set were congealed and used to form a funnel to subsequently align all of the images in both the training and test sets. To obtain verification accuracy, use a variation on the method of Cosine Similarity Metric Learning (CSML), one of the top-performing methods on LFW. As in CSML, first apply whitening PCA and reduce the representation to 500 dimensions. Then normalize each image feature vector, and apply a linear SVM to an image pair by combining the image feature vectors using element-wise multiplication. Note that if the weights of the SVM are 1 and the bias is 0, then this is equivalent to cosine similarity. As the goal is to improve verification accuracy through better alignment, focus on performance using a single feature representation, and only use the square root LBP features on 150x80 croppings of the full LFW images.

[00150] Table 8 gives the verification accuracy for this verification system using images produced by a number of alignment algorithms. Deep congealing gives a significant improvement over SIFT congealing. Using a CDBN representation learned with a group sparsity penalty, leading to learned filters with topographic organization, consistently gives a higher accuracy of one to two percentage points. Congealing with a one-layer CDBN (technically speaking, the term "one-layer CDBN" denotes a CRBM) trained on faces, with topology, gives verification accuracy significantly higher than conventional approaches and comparable to the accuracy using LFW-a images.

Table 8

Alignment	Accuracy
Original	0.742
SIFT Congealing	0.758
Deep Congealing, Kyoto, layer 1	0.807
Deep Congealing, Kyoto, layer 1, with topology	0.815
Deep Congealing, Faces, layer 1	0.802
Deep Congealing Faces, layer 1, with topology	0.820
Deep Congealing, Faces, layer 2	0.780
Deep Congealing, Faces, layer 2, with topology	0.797
Combining Scores of Faces, layer 1 and 2, with topology	**0.831**
Fiducial Points-based Alignment [3] (supervised)	0.805
LFW-a (commercial)	0.823

[00151] Moreover, the verification scores can be combined using images from the one-layer and two-layer CDBN trained on faces, learning a second SVM on these scores. By doing so, a further gain is achieved in verification performance, achieving an accuracy of 0.831, exceeding the accuracy using LFW-a. This suggests that the two-layer CDBN alignment is somewhat complementary to the one- layer alignment. That is, although the two-layer CDBN alignment produces a lower verification accuracy, it is not strictly worse than the one-layer CDBN alignment for all images, but rather is aligning according to a different set of statistics, and achieves success on a different subset of images than the one-layer CDBN model. As a control, the same score combination is performed using the scores produced from images from the one-layer CDBN alignment trained on faces, with topology, and the original images. This gave a verification accuracy of 0.817, indicating that the improvement from combining two-layer scores is not merely obtained from using two different sets of alignments.

[00152] It has been shown how to combine unsupervised joint alignment with unsupervised feature learning. By congealing on the pooling layer representation of a CDBN, significant gains are achieved in verification accuracy over existing methods for unsupervised alignment. By adding a group sparsity penalty to the CDBN learning algorithm, filters can be learned with a linear topology, providing a smoother optimization landscape for congealing. Using face images aligned by this method, higher verification accuracy is obtained than the supervised fiducial points based method. Further, despite being unsupervised, this method is still able to achieve comparable accuracy to the widely used LFW-

a images, obtained by a commercial fiducial point-based alignment system whose detailed procedure is unpublished.

[00153] There has been a significant amount of progress made in the area of face recognition, with recent research focusing on the face verification problem. In this set-up, pairs of images are given at training time, along with a label indicating whether the pair contains two images of the same person (matched pair), or two images of two different persons (mismatched pair). At test time, a new pair of images is presented, and the task is to assign the appropriate matched/mismatched label. Unlike other face recognition problem formulations, it is not assumed that the person identities in the training and test sets have any overlap, and often the two sets are disjoint.

[00154] This set-up removes one of the fundamental assumptions of the traditional experimental design, making it possible to perform recognition on never-before-seen faces. Another important assumption that has been relaxed recently is the amount of control the experimenter has over the acquisition of the images. In unconstrained face verification, the only assumption made is that the face images were detected by a standard face detector. In particular, images contain significant variations in nuisance factors such as complex background, lighting, pose, and occlusions. These factors lead to large intra-class differences, making the unconstrained face verification problem very difficult.

[00155] The current standard for benchmarking performance on unconstrained face verification is the Labeled Faces in the Wild (LFW) data set. Since the release of the database, classification accuracy on LFW has improved dramatically, from initial methods getting less than 0.75 accuracy to current state-of-the-art methods getting 0.84 to 0.86 accuracy.

[00156] The majority of existing methods for face verification rely on feature representations given by hand-crafted image descriptors, such as SIFT and Local Binary Patterns (LBP). Further performance increases are obtained by combining several of these descriptors. Rather than spending time attempting to engineer new image descriptors by hand, instead it is proposed to obtain new representations automatically through unsupervised feature learning with deep network architectures.

[00157] These representations offer several advantages over those obtained through hand-crafted descriptors: They can capture higher-order statistics such as corners and contours, and can be tuned to the statistics of the specific object classes being considered (*e.g.,* faces). Further, an end system making use of deep learning features can be more readily adapted to new domains where the hand-crafted descriptors may not be appropriate.

[00158] The convolutional restricted Boltzmann machine is an extension of the restricted Boltzmann machine (RBM). The RBM is a Markov random field with a hidden layer and a visible layer (corresponding to input data, such as image pixels), with bipartite connections between the layers *(i.e.,* there are no connections among visible nodes or among hidden nodes). In a convolutional restricted Boltzman machine (CRBM), rather than fully connecting the hidden layer and visible layer, the weights between the hidden units and the visible units are local (*i.e.,* 10x10 pixels instead of full image) and shared among all locations in the hidden units. The CRBM captures the intuition that if a certain image feature (or pattern) is useful in some locations of the image, then the same image feature can also be useful in other locations.

[00159] In this disclosure, a convolutional RBM is utilized with real-valued visible input nodes v and binary-valued hidden nodes h. The visible input nodes can be viewed as intensity values in the $N_V \times N_V$ pixel image, and the hidden nodes are organized in 2-D configurations (*i.e.*, $\mathrm{v} \in \mathbb{R}^{\mathrm{N_v} \times \mathrm{N_v}}$ and $h \in \{0,1\}^{N_H \times N_H}$). An illustration of an CRBM can be found in Figure 15.

[00160] The CRBM has three sets of parameters: (1) K convolution filter weights between a hidden node and the visible nodes, where each filter covers N_W x N_W pixels (*i.e.,* $W^k \in \mathbb{R}^{Nw \times Nw}$, $k = 1, \ldots, K$); (2) hidden biases $b^k \in \mathbb{R}$ that are shared among hidden nodes; and (3) visible bias $c \in \mathbb{R}$ that is shared among visible nodes.

[00161] To make CRBMs more scalable, probabilistic max-pooling is a technique for incorporating local translation invariance. Max-pooling refers to operations where a local neighborhood (*e.g.,* 2x2 grid) of feature detection outputs is shrunk to a pooling node by computing the maximum of the local neighbors. Max-pooling makes the feature representation become more invariant to local translations in the input data, and it has been shown to be useful in visual

recognition problems. Probabilistic max-pooling enables the CRBM to incorporate max-pooling like behavior, while allowing probabilistic inference (such as bottom-up and top-down inference). It further enables increasingly more invariant representations as we stack CRBMs.

[00162] The energy function of the probabilistic max-pooling CRBM (with real-valued visible units) is defined as follows:

$$
\begin{aligned}
P(\mathbf{v},\mathbf{h}) &= \frac{1}{Z}\exp(-E(\mathbf{v},\mathbf{h})) \\
E(\mathbf{v},\mathbf{h}) &= -\sum_{k=1}^{K}\sum_{i,j=1}^{N_H}\sum_{r,s=1}^{N_W} h_{ij}^k W_{rs}^k v_{i+r-1,j+s-1} \\
&\quad + \sum_{i,j=1}^{N_V}\frac{1}{2}v_{ij}^2 - \sum_{k=1}^{K} b_k \sum_{i,j=1}^{N_H} h_{ij}^k - c\sum_{i,j=1}^{N_V} v_{ij} \\
\text{s.t.} &\quad \sum_{(i,j)\in B_\alpha} h_{ij}^k \le 1, \quad \forall k,\alpha \qquad (1)
\end{aligned}
$$

Here, B_α refers to a C x C block of locally neighboring hidden units h_{ij}^k that are pooled to a pooling node p_α^k.

[00163] Under this energy function, the conditional probabilities can be computed as follows:

$$P(v_{ij}=1|\mathbf{h}) = \mathcal{N}\left(\left(\sum_k W^k *_f h^k\right)_{ij} + c, 1\right) \quad (2)$$

$$P(h_{ij}^k=1|\mathbf{v}) = \frac{\exp(I(h_{ij}^k))}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \quad (3)$$

where $I(h_{ij}^k) \triangleq b_k + (\tilde{W}^k *_v v)_{ij}$, $\mathcal{N}(\cdot)$ is a normal distribution, $\tilde{W}$ refers to flipping the original filter W in both upside-down and left-right directions, $*_v$ denotes valid convolution, and $*_f$ denotes full convolution.

[00164] At the same time, the pooling node p_α^k is a stochastic random variable that is defined as $p_\alpha^k \triangleq \sum_{(i,j)\in B_\alpha} h_{ij}^k$ and the marginal posterior can be written as a softmax function:

$$P(p_\alpha^k=1|\mathbf{v}) = \frac{\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))}{1+\sum_{(i',j')\in B_\alpha}\exp(I(h_{i',j'}^k))} \quad (4)$$

When sampling from the posterior (given the visible nodes), the hidden nodes in each block can be efficiently sampled in parallel from multinomial distributions, then set the pooling node values accordingly.

[00165] The objective function is the log-likelihood of the training data. Although exact maximum likelihood training is intractable, the contrastive divergence approximation allows us to estimate an approximate gradient efficiently. Contrastive divergence is not unbiased, but has low variance, and has been successfully applied in optimizing many undirected graphical models that have intractable partition functions.

[00166] Sparsity regularization is applied. Since the model is highly over-complete, it is necessary to regularize the model to prevent it from learning trivial or uninteresting feature representations. Specifically, add a sparsity penalty term to the log-likelihood objective to encourage each hidden unit group to have a mean activation close to a small constant. This was implemented this with the following simple update rule (following each contrastive divergence update):

$$\Delta b_k \propto p - \frac{1}{N_H^2} \sum_{i,j} P(h_{ij}^k = 1|\mathbf{v}), \qquad (5)$$

where p is a target sparsity, and each image is treated as a mini-batch. The learning rate for the sparsity updates was chosen to make the hidden group's average activation (over entire training data) close to the target sparsity, while allowing variations of activations depending on specific input images.

[00167] After training a max-pooling CRBM, use it to compute the posterior of the hidden (pooling) units given the input data. These hidden (pooling) unit "activations" can be used as input to further train the next layer CRBM.

[00168] By stacking the CRBMs, the algorithm can capture high-level features, such as hierarchical object-part decompositions. In experiments, CDBNs were trained with up to two layers of CRBMs. After constructing a convolutional deep belief network, perform (approximate) inference of the whole network in a feedforward (bottom-up) manner.

[00169] The weight sharing scheme in a CRBM assumes that the distribution over features is stationary in an image with respect to location. However, for images belonging to a specific object class, such as faces, this assumption is no longer true. One strategy for removing this stationarity assumption is to connect each hidden unit to only a local receptive field in the

visible image, as in the CRBM, but remove the parameter tying between weights for different hidden units. However, even with only local connections, without any parameter tying, it is computationally intractable to scale this model to high resolution images (*e.g.*, 150x150 pixel images in the LFW dataset). Moreover, without parameter tying, the model becomes sensitive to local deformations and misalignments.

[00170] To maintain the advantages of the CRBM while exploiting global structure, divide the image into a number of overlapping regions. The *local convolutional restricted Boltzmann machine* extends the CRBM by using a separate set of weights for each region. When trained on images with some global structure, a local CRBM can learn a more efficient representation than a CRBM since features for a particular location are learned only if they are useful for representing the corresponding region. Moreover, since filter weights are no longer shared globally, a local CRBM may be able to avoid spurious activations of hidden units outside the pre-specified local regions.

[00171] The local CRBM is formulated as follows. First, divide the image into L overlapping regions, with the l-th region defined as $\left\{R^{(l)} : \left(r_{min}^{(l)}, r_{max}^{(l)}, c_{min}^{(l)}, c_{max}^{(l)}\right)\right\}$, where r and c represent row or column index for the region in the image. For convenience of presentation, assume that each region is square, with height and width equal to N_R. Denote by $\mathrm{v}^{(l)}$ the "submatrix" of the visible units that correspond to the l-th region.

[00172] Let each region have K filters $W^{(l),k}$ of size N_W x N_W. The hidden units $h^{(l),k}$ are binary random variables with 2D spatial structure ($N_H \times N_H$ grid), where $N_H \triangleq N_R - N_W + 1$. The energy function of the local convolutional RBM is now defined as follows:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{l=1}^{L}\sum_{k=1}^{K}\left(\mathbf{v}^{(l)} * \tilde{W}^{(l),k}\right) \odot \mathbf{h}^{(l),k} \qquad (6)$$
$$+ \sum_{i,j=1}^{N_V} \frac{1}{2}(\mathbf{v}_{ij} - c)^2 + \sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{r,s=1}^{N_H} b_k^{(l)} \mathbf{h}_{r,s}^{(l),k}$$

where $\odot$ is the element-wise product operator, c is a visible bias, and $b_k^{(l)}$ is a hidden bias. With v fixed, the conditional probability of hidden units can be defined as:

$$P(\mathbf{h}_{r,s}^{(l),k} = 1|\mathbf{v}^{(l)}) = \sigma((\mathbf{v}^{(l)} * \tilde{W}^{(l),k})_{r,s} + b_k^{(l)}). \quad (7)$$

where the $\sigma(x) = \frac{1}{1+\exp(-x)}$. Also, define the conditional probability of the visible units given the hidden units.

$$P(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\sum_{l=1}^{L} I^{(l)}\left(\sum_{k=1}^{K} W^{(l),k} *_f \mathbf{h}^{(l),k}\right) + c, \mathbf{I}\right). \quad (8)$$

Here, $h = \{h^{(1)}, \ldots, h^{(L)}\}$, and $I^{(l)}(Y)$ is a projection operator from $R^{N_R \times N_R}$ to $R^{N_v \times N_v}$ where Y is a $N_R \times N_R$ image, used to accumulate the contribution of each local region to the visible layer. Specifically, $I^{(l)}(Y)$ is defined as

$$\left[I^{(l)}(Y_{r',c'})\right]_{r,c} = \begin{cases} Y_{r',c'} & \text{if } (r,c) = \\ & (r' + r_{min}^{(l)} - 1, c' + c_{min}^{(l)} - 1) \\ 0 & \text{otherwise.} \end{cases}$$

[00173] With these conditional probabilities, train the local CRBM following the similar procedure as for the CRBM using contrastive divergence. It is noted that the probabilistic max-pooling is defined for the local CRBM. However, for the simplicity of presentation, a case without probablistic max-pooling is presented. Further, note that binary local CRBM is used when the local CRBM is stacked as the second layer.

[00174] Deep learning for images is usually performed by letting the visible units be whitened pixel intensity values. Additional novel representations are learned by learning deep networks on Local Binary Patterns, demonstrating the potential for learning representations that capture higher-order statistics of hand-crafted image descriptors. Using uniform LBPs (at most two bitwise transitions), a 59 dimensional binary vector is at each pixel location. Find a small increase in performance by first forming histograms of 3x3 neighbors (average pooling), and then learning a binary CRBM on this representation.

[00175] Inspired by the success of Cosine Similarity Metric Learning (CSML), the face verification algorithm is also based on a metric-learning approach. For the handcrafted model, use the same features as was used with CSML (pixel intensity, LBP, Gabor). Additionally follow the same set-up by using principle component analysis (PCA) to reduce the dimensionality to 500, for all feature representations.

[00176] Rather than using CSML to learn a matrix A_{CSML}, instead apply Information-Theoretic Metric Learning (ITML) to produce a Mahalanobix matrix M. Then perform a Cholesky decomposition yielding a matrix A such that $A^T A = M$.

[00177] Letting x be the representation of an image after applying PCA, obtain a feature vector y for an image by applying A and unit-normalizing, $y = \frac{A_x}{\|A_x\|}$. Then form a feature vector z for a pair of two images (with features y and y', respectively) using element-wise multiplication $z = y \odot y'$. Finally, apply a linear SVM to the feature vectors z (for pairs of images) to perform face verification.

[00178] In practice, using ITML improves performance over CSML by several percentage points. Note that if A is the identity matrix and the weights of the SVM are 1, then the system reduces to cosine similarity. Consistent with previous work, compression using PCA followed by normalization gave the best performance.

[00179] For experiments, the LFW-a face images are aligned using a commercial face alignment software. Use three croppings of each image (150x150, 125x75, 100x100), resizing to the same input size for the visible layer, to capture information at different scales. For self-taught learning, images were used from the Kyoto natural images data set (found at http://www.cnbc.crnu.edu/cplab/data_kyoto.html).

[00180] To solve the SVM, the Shogun Toolbox is used (found at http://www.shogun-toolbox.org/). Set the SVM C parameter using the development view of LFW. The CDBN code was optimized to use a GPU (e.g., code from Graham Taylor http://www.cs.nyu.edu/~gwtaylor/code/GPUmat/), allowing us to test a single kernel system in several minutes and learn weights in a DBN in less than an hour.

[00181] One of the challenges of using a deep learning architecture is the number of architecture and model hyperparameters that one must set. For each layer of CDBN, decide the size of the filters, number of filters, max-pooling region size, and sparsity of the hidden units.

[00182] Saxe et al's in "On Random Weights and Unsupervised Feature Learning", ICML, 2011, found some correlation between performance with random filters and learned filters for a given architecture, and suggested using search over architectures with random filters as a proxy for selecting a best architecture to use with learned weights.

[00183] The correlation between random weight and learned weight performance is evaluated for a one layer network with 16 different architectures, varying the above architecture hyperparameters. In this experiment, a single cropping only is used and not metric learning. Figure 16 shows a scatter plot of random weight performance versus learned weight performance. Note a somewhat high correlation of 0.40. However, a more interesting finding is that the range of accuracies for the learned filters is much more concentrated around higher values compared with the random filters. Thus, it can be hypothesized that, although networks with random filters can approach the same accuracy as networks with learned filters given the right architecture, an added benefit of learning is that the accuracy becomes more robust to the specific architecture hyperparameters.

[00184] Moreover, find that multi-layer networks with random weights at each layer yield representations that lead to near-chance recognition performance. Empirically, this seems to indicate that, at least for the face verification task, the non-linearities in a multi-layer network with random filters do not give good representations, and learning is necessary. Given these findings, set the hyperparameters by performing a coarse search over the possible values, and learning and evaluating the model on the development view of LFW.

Table 9

Source	Rep.	Layer	Model	Accuracy
Kyoto	pixels	1	CRBM	0.8527
Faces	pixels	1	CRBM	0.8530
Kyoto	pixels	2	CRBM	0.8522
Faces	pixels	2	CRBM	0.8457
Faces	pixels	2	local CRBM	0.8538
Kyoto	LBP	1	CRBM	0.8520
Faces	LBP	1	CRBM	0.8485
Kyoto	pixels	1+2		0.8572
Faces	pixels	1+2		0.8582
Kyoto	both	1+2		0.8660
Faces	both	1+2		0.8642
both	both	1+2		0.8688

[00185] The top section of Table 9 gives the accuracy for individual deep architectures. Since the basic image features learned by a single layer CRBM are expected to be largely edgelike features that are shared throughout the image, apply the local CRBM model only at the second layer. The second layer CRBM and local CRBM have approximately the same size hidden layer representation, but the local CRBM is able to learn more filters since they are specific to each region, and achieves a higher accuracy. The bottom section of Table 1 gives the accuracy when combining the scores from multiple deep architectures using a linear SVM. As the different layers are capturing complementary information, higher accuracy is achieved by fusing these scores.

[00186] Table 10 gives the final accuracy of the proposed system using the deep learning representations, and the combined deep learning and hand-crafted image descriptor representations, in comparison with other systems trained using the image-restricted setting of LFW. The system, using only deep learning representations, is competitive with state-of-the-art methods that rely on a combination of descriptions of hand-crafted image descriptors, and is state-of-the-art relative to the existing deep learning method, despite the fact that it used manual annotations of eye coordinates to align the faces.

Table 10

Method	$\hat{\mu} \pm S_E$
V1-like with MKL [25]	0.7935 ± 0.0055
Linear rectified units [19]	0.8073 ± 0.0134
CSML [20]	0.8418 ± 0.0048
Learning-based descriptor [3]	0.8445 ± 0.0046
OSS, TSS, full [38]	0.8683 ± 0.0034
OSS only [38]	0.8207 ± 0.0041
Hand-crafted	0.8718 ± 0.0049
Deep Learning	0.8688 ± 0.0062
Combined	$\mathbf{0.8777 \pm 0.0062}$

[00187] Additional insight into the face verification problem is gained by looking at the number of representations whose score correctly classifies each pair. Figure 17 shows a histogram over these values, separately for mismatched pair and matched pairs. Interestingly, the pairs that are correctly classified by few or no representations is heavily skewed toward matched pairs. These image pairs highlight a fundamental difficulty with face verification, and verification within an object class in general, namely the large amount of intra-class variation due to several factors (e.g., pose).

[00188] In recent years, unsupervised feature learning algorithms have emerged as promising tools for learning representations from data. In particular it is an important problem to learn invariant representations that are robust to variability in high-dimensional data (e.g., images, speech, etc.) since they will enable machine learning systems to achieve good generalization performance while using a small number of labeled training examples. In this context, several feature learning algorithms have been proposed to learn invariant representations for specific transformations by using customized approaches. For example, convolutional feature learning methods can achieve shift-invariance by exploiting convolution operators. As another example, the denoising autoencoder can learn features that are robust to the input noise by trying to reconstruct the original data from the hidden representation of the perturbed data. However, learning invariant representations with respect to general types of transformations is still a challenging problem.

[00189] In yet another aspect of this disclosure, a novel framework of transformation-invariant feature learning is presented. Local transformations (e.g., small amounts of translation, rotation, and scaling in images) are the focus, which can be approximated as linear transformations, and incorporate linear transformation operators into the feature learning algorithms. For example, the transformation-invariant restricted Boltzmann machine is presented, which is a generative model that represents input data as a combination of transformed weights. In this case, a transformation-invariant feature representation is obtained via probabilistic max pooling of the hidden units over the set of transformations. In addition, we show extensions of our transformation-invariant feature learning framework to other unsupervised feature learning algorithms, such as autoencoders or sparse coding.

[00190] A general framework is first presented for learning locally-invariant features using transformations. For illustration purposes, the restricted Boltzmann machine (RBM) is used as the main example although other type of neural network are also contemplated. The restricted Boltzmann machine is a bipartite undirected graphical model that is composed of visible and hidden layers. Assuming binary-valued visible and hidden units, the energy function and the joint probability distribution are given as follows:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{v}^T\mathbf{W}\mathbf{h} - \mathbf{b}^T\mathbf{h} - \mathbf{c}^T\mathbf{v}, \qquad (1)$$

$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp\left(-E(\mathbf{v},\mathbf{h})\right) \qquad (2)$$

where $\boldsymbol{v} \in \{\mathbf{0},\mathbf{1}\}^D$ are binary visible units, $\boldsymbol{h} \in \{\mathbf{0},\mathbf{1}\}^K$ are binary hidden units, and $\boldsymbol{W} \in \mathbb{R}^{D\times K}$, $\boldsymbol{b} \in \mathbb{R}^K$, and $\boldsymbol{c} \in \mathbb{R}^D$ are weights, hidden biases, and visible biases, respectively. Z is a normalization factor that depends on the parameters **{W, b, c}.** Since RBMs have no intra-layer connectivity, exact inference is tractable and block Gibbs sampling can be done efficiently using the following conditional probabilities:

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(\sum_i v_i W_{ij} + b_j\right) \qquad (3)$$

$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}\left(\sum_j h_j W_{ij} + c_i\right) \qquad (4)$$

where $\text{sigmoid}(x) = \frac{1}{1+exp(-x)}$. The RBM parameters are trained by minimizing the negative log-likelihood via stochastic gradient descent. Although computing the exact gradient is intractable, it can be approximated using contrastive divergence. Due to space constraints, only the case of binary-valued input variables is presented; however, the RBM with real-valued input variables can be formulated straightforwardly.

[00191] Next, a novel feature learning framework is formulated that can learn invariance to a set of linear transformations based on the RBM. The transformation operator is defined as a mapping $T: \mathbb{R}^{D_1} \rightarrow \mathbb{R}^{D_2}$ that maps D_1-dimensional input vectors into D_2-dimensional output vectors *($D_1 \geq D_2$)*. In this case, assume a linear transformation matrix $T \in \mathbb{R}^{D_2 \times D_1}$ i.e., each coordinate of the output vector is represented as a linear combination of the input coordinates.

[00192] With this notation, formulate the *transformation- invariant restricted Boltzmann machine* (TIRBM) that can learn invariance to a set of transformations. Specifically, for a given set of transformation matrices T_s $(s = 1, \ldots, S)$, the energy function of TIRBM is defined as follows:

$$E(\mathbf{v}, \mathbf{H}) = \quad (5)$$
$$-\sum_{j=1}^{K}\sum_{s=1}^{S}(T_s\mathbf{v})^T\mathbf{w}_j h_{j,s} - \sum_{j=1}^{K}\sum_{s=1}^{S} b_{j,s}h_{j,s} - \mathbf{c}^T\mathbf{v}$$
$$\text{s.t.} \sum_{s=1}^{S} h_{j,s} \leq 1, h_{j,s} \in \{0,1\}, j = 1, \cdots, K, \quad (6)$$

where **v** are D_1-dimensional visible units, and $\mathbf{w}_j$ are D_2-dimensional (filter) weights corresponding to the *j*-th hidden unit. The hidden units are represented as a matrix $H \in \{0,1\}^{K \times S}$ with $h_{j,s}$ as its (j,s)-th entry. In addition, denote $z_j = \sum_{s=1}^{S} h_{j,s} z_j \in \{0,1\}$ as a pooled hidden unit over the transformations.

[00193] In Equation (6), impose a softmax constraint on hidden units so that at most one unit is activated at each row of H. This probabilistic max pooling allows one to obtain a feature representation invariant to linear transformations. More precisely, suppose that the input v_1 matches the filter w_j. A similar technique is used in convolutional deep belief networks, in which spatial probabilistic max pooling is applied over a small spatial region. Given another input v_2 that is a transformed version of v_1, the TIRBM will try to find a

transformation matrix T_{sj} so that the v_2 matches the transformed filter $T_{s_j}^T w_j$ i.e.,$v_1^T w_j \approx v_2^T T_{s_j}^T w_j$. Note that the transpose T_s^T of a transformation matrix T_s also induces a linear transformation. Therefore, v_1 and v_2 will both activate z_j after probabilistic max pooling. Figure 18 illustrates this idea.

[00194] Compared to the regular RBM, the TIRBM can learn more diverse patterns, while keeping the number of parameters small. Specifically, multiplying transformation matrix (e.g., $T_s^T w_j$) can be viewed as increasing the number of filters by the factor of S, but without significantly increasing the number of parameters due to parameter sharing. In addition, by pooling over local transformations, the filters can learn invariant representations (i.e., z_j's) to these transformations.

[00195] The conditional probabilities are computed as follows:

$$p(h_{j,s} = 1|\mathbf{v}) = \frac{\exp(\mathbf{w}_j^T T_s \mathbf{v} + b_{j,s})}{1 + \sum_{s'} \exp(\mathbf{w}_j^T T_{s'} \mathbf{v} + b_{j,s'})} \quad (7)$$

$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}(\sum_{j,s} (T_s^T \mathbf{w}_j)_i h_{j,s} + c_i) \quad (8)$$

Similar to RBM training, stochastic gradient descent is used to train TIRBM. The gradient of the log-likelihood is approximated via contrastive divergence by taking the gradient of the energy function (Equation (5)) with respect to the model parameters.

[00196] The sparseness of the feature representation is often a desirable property. By following Lee et al's approach, the model can be extended to sparse TIRBM by adding the following regularizer for a given set of data $\{v^{(1)}, \ldots, v^{(N)}\}$ to the negative log-likelihood:

$$\mathcal{L}_{sp} = \sum_{j=1}^{K} \mathcal{D}\left(p, \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[z_j^{(n)}|\mathbf{v}^{(n)}]\right) \quad (9)$$

where $\mathcal{D}$ is a distance function; p is the target sparsity. The expectation of pooled activation is written as

$$\mathbb{E}[z_j|\mathbf{v}] = \frac{\sum_s \exp(\mathbf{w}_j^T T_s \mathbf{v} + b_{j,s})}{1 + \sum_s \exp(\mathbf{w}_j^T T_s \mathbf{v} + b_{j,s})}. \quad (10)$$

Note that regularize over the pooled hidden units z_j rather than individual hidden units $h_{j,s}$. In experiments, L2 distance is used for $\mathcal{D}(\cdot,\cdot)$, but one can also use KL divergence for the sparsity penalty.

[00197] The design the transformation matrix T is described. For the ease of presentation, assume 1-d transformations, but it can be extended to 2-d cases (e.g., image transformations) straightforwardly. Further, assume the case of $D_1 = D_2 = D$ here, but, we will discuss general cases later.

[00198] As mentioned previously, $T \in \mathbb{R}^{D\times D}$ is a linear transformation matrix from x $\in \mathbb{R}^D$ *to* y $\in \mathbb{R}^D$; i.e., each coordinate of y is constructed via linear combination of the coordinates in x with weight matrix T as follows:

$$y_i = \sum_{j=1}^{D} T_{ij} x_j, \forall i = 1, \cdots, D. \qquad (11)$$

For example, shifting by s can be defined as

$$T_{ij}(s) = \begin{cases} 1 & \text{if } i = j + s, \\ 0 & \text{otherwise.} \end{cases}$$

For 2-d image transformations such as rotation and scaling, the contribution of input coordinates to each output coordinate is computed with bilinear interpolation. Since $T's$ are pre-computed once and usually sparse, Equation (11) can be computed efficiently.

[00199] The transformation-invariant feature learning framework is not limited to the energy-based probabilistic models, but can be extracted to other unsupervised learning methods as well. For example, it can be readily adapted to autoencoders by defining the following softmax encoding sigmoid decoding functions:

$$f_{j,s}(\mathbf{v}) = \frac{\exp\left(\mathbf{w}_j^T T_s \mathbf{v} + b_{j,s}\right)}{1 + \sum_{s'} \exp\left(\mathbf{w}_j^T T_{s'} \mathbf{v} + b_{j,s'}\right)} \qquad (12)$$

$$\hat{v}_i = \text{sigmoid}\left(\sum_{j,s} (T_s^T \mathbf{w}_j)_i f_{j,s} + c_i\right) \qquad (13)$$

[00200] Following the idea of TIRBM, the transformation-invariant sparse coding can be formulated as follows:

$$\min_{\mathbf{W},\mathbf{H}} \sum_{n} \| \sum_{j=1}^{K} \sum_{s=1}^{S} T_s^T \mathbf{w}_j h_{j,s}^{(n)} - \mathbf{v}^{(n)} \|^2, \quad (14)$$

$$\text{s.t. } \|\mathbf{H}\|_0 \leq \gamma, \|\mathbf{H}(j,:)\|_0 \leq 1, \|\mathbf{w}_j\|_2 \leq 1, \quad (15)$$

where γ is a constant. The second constraint in (15) can be understood as an analogy to the softmax constraint in Equation (6) of TIRBMs.

[00201] Similar to standard sparse coding, the parameters can be optimized by alternately optimizing **W** and **H** while fixing the other. Specifically, **H** can be (approximately) solved using Orthogonal Matching Pursuit, and therefore we refer this algorithm a transformation-invariant Orthogonal Matching Pursuit (TIOMP).

[00202] For images, assume a receptive field size of $r \times r$ pixels (for input image patches) and a filter size of $w \times w$ pixels. Define gs to denote the number of pixels corresponding to the transformation (e.g., translation or scaling). For example, translate the $w \times w$ filter across the $r \times r$ receptive field with a stride of gs pixels (Figure 19A), or scale down from $(r - l \cdot gs) \times (r - l \cdot gs)$ to $w \times w$ (where $0 \leq l \leq \left\lfloor \frac{r-w}{gs} \right\rfloor$) by sharing the same center for the filter and the receptive field (Figure 19B).

[00203] For classification tasks, the posterior probability of the pooled hidden unit (Equation (10)) is used as a feature. Note that the dimension of the extracted feature vector for each image patch is K, not $K \times S$. Thus, argue that the performance gain of the TIRBM over the regular RBM comes from the better representation (i.e., transformation-invariant features), rather than from the classifier's use of higher-dimensional features.

[00204] First, the performance of the proposed algorithm is verified on the variations of a handwritten digit dataset, assuming that the transformation information is given. From the MNIST variation datasets, tests were performed on "mnist-rot" (rotated digits, referred to as rot) and "mnist-rot-back-image" (rotated digits with background images, referred to as rot-bgimg). To further evaluate with different types of transformations, create four additional datasets that contain scale and translation variations with and without random background (referred to as scale, scale-bgrand, trans, and trans-bgrand, respectively). Some examples are shown in Figure 20.

[00205] For these datasets, sparse TIRBMs on the image patches of size 28×28 pixels with dataspecific transformations are trained. For example, consider 16 equally-spaced rotations (i.e., the step size of $\frac{\pi}{8}$) for the rot and rot-bgimg datasets. Similarly, for the scale and scale-bgrand datasets, scale-transformation matrices with $w = 20$ and $gs = 2$ are generated, which can map from $(28 - 2l) \times (28 - 2l)$ pixels to 20×20 pixels with $l \in \{0, \ldots, 4\}$. For the trans and trans-bgrand datasets, set $w = 24$ and $gs = 2$ to have total nine translation matrices that cover the 28×28 regions using 24×24 pixels with a horizontal and vertical stride of 2 pixels. For classification, 1,000 filters were trained for both sparse RBMs and sparse TIRBMs and used a softmax classifier. 10,000 examples were used for the training set, 2,000 examples for the validation set, and 50,000 examples for the test set.

Table 11

Dataset	RBM	TIRBM	Transformation
rot	15.6%	4.2%	rotation
rot-bgimg	54.0%	35.5%	rotation
scale	6.1%	5.5%	scaling
scale-bgrand	32.1%	23.7%	scaling
trans	15.3%	9.1%	translation
trans-bgrand	57.3%	43.7%	translation

[00206] As reported in Table 11, the proposed method (sparse TIRBMs) consistently outperformed the baseline method (sparse RBMs) for all datasets. These results suggest that the TIRBMs can learn better representations for the foreground objects by transforming the filters. It is worth noting that the error rates for the mnist-rot and mnist-rot-back-image datasets are also significantly lower than the best published results obtained with stacked denoising autoencoders (i.e., 9.53% and 43.75%, respectively).

[00207] For qualitative evaluation, the learned filters are visualized on the mnist-rot dataset trained with the sparse TIRBM (Figure 20E) and the sparse RBM (Figure 20F), respectively. The filters learned from sparse TIRBMs show much clearer pen-strokes than those learned from sparse RBMs, which partially explains the impressive classification performance.

[00208] For handwritten digit recognition in the previous section, assume prior information on global transformations on the image (e.g.,

translation, rotation and scale variations) for each dataset. This assumption enabled the proposed TIRBMs to achieve significantly better classification performance than the baseline method, since the data-specific transformation information was encoded in the TIRBM.

[00209] However, for natural images, it is not reasonable to assume such global transformations due to the complex image structures. In fact, recent literature suggests that some level of invariance to local transformations (e.g., few pixel translation or coordinate-wise noise) leads to improved performance in classification. From this viewpoint, it makes more sense to learn representations with local receptive fields that are invariant to generic image transformations (e.g., small amounts of translation, rotation, and scaling), which does not require data-specific prior information.

[00210] The learned TIRBM filters are visualized in Figure 21, where the 14×14 natural image patches taken from the van Hateren dataset. The baseline model (sparse RBM) learns many similar vertical edges (Figure 21A) that are shifted by a few pixels, whereas the proposed methods can learn diverse patterns, including diagonal and horizontal edges, as shown in Figure 21B, 21C, and 21D. These results suggest that TIRBMs can learn diverse sets of filters, which is reminiscent of the effects of convolutional training. However, this model is much easier to train than convolutional models, and it can further handle generic transformations beyond translations.

[00211] Image classification tasks were also evaluated using two datasets. First, the widely used CIFAR-10 dataset was tested, which is composed of 50,000 training and 10,000 testing examples with 10 categories. Rather than learning features from the whole image (32×32 pixels), TIRBMs were trained on local image patches while keeping the RGB channels. A fixed filter size $w = 6$ was used and the receptive field size was determined depending on the types of transformations. For example, $r = 6$ for rotations. For both scale variations or translations, $r = 8$ and $gs = 2$. Then, after unsupervised training with TIRBM, the convolutional feature extraction scheme is used. Specifically, the TIRBM pooling-unit activations are computed for each local $r \times r$ pixel patch that was densely extracted with a stride of 1 pixel, and averaged the patch-level activations over each of the 4 quadrants in the image. Eventually, this procedure

yielded $4K$-dimensional feature vectors for each image, which were fed into an L2-regularized linear SVM. 5-fold cross validation was performed to determine the hyperparameter C.

[00212] For comparison to the baseline model, the sparse TIRBMs with a single type of transformation (translation, rotation, or scaling) was separately evaluated using $K = 1{,}600$. As shown in Table 12, each single type of transformation in TIRBMs brought a significant performance gain over the baseline sparse RBMs. The classification performance was further improved by combining different types of transformations into a single model.

Table 12

Methods (Linear SVM)	Accuracy
sparse RBM†(baseline)	72.4%
sparse TIRBM (scaling only)	76.8%
sparse TIRBM (rotation only)	77.7%
sparse TIRBM (translation only)	77.5%
sparse TIRBM (combined)	78.8%
sparse TIRBM (combined, K=4,000)	80.1%
TIOMP-1/T (combined)	80.7%
TIOMP-1/T (combined, K-4,000)	82.2%
VQ (K=4,000)†	79.6%
OMP-1/T (K=1,600)‡	79.4%
OMP-1/T (K=6,000)‡	81.5%
convolutional DBN (Krizhevsky, 2010)	78.9%
deep NN (Ciresan et al., 2011)	80.5%
deep NN (Coates & Ng, 2011b)	82.0%

[00213] In addition, the classification results obtained using TIOMP-1 for unsupervised training are reported. In this experiment, the following two-sided soft thresholding encoding function is used:

$$f_j = \max_S \{\max(\mathrm{w}_j^T T_s \mathrm{v}\text{-}\alpha, 0)\}$$

$$f_{j+K} = \max_S \{\max(-w_j^T T_s \mathrm{v}\text{-}\alpha, 0)\}$$

where α is a constant threshold that was cross-validated. As a result, it was observed about 1% improvement over the baseline method (OMP-1/T) using 1,600 filters, which supports the argument that our transformation-invariant feature learning framework can be effectively transferred to other unsupervised learning methods. Finally, by increasing the number of filters ($K = 4{,}000$), better

results (82.2%) were obtained than the previously published results using single-layer models, as well as those using deep networks.

Table 13

Methods (Linear SVM)	Acc. ± std.
sparse RBM	55.0±0.5%
sparse TIRBM (scaling only)	55.9±0.7%
sparse TIRBM (rotation only)	57.0±0.7%
sparse TIRBM (translation only)	57.8±0.5%
sparse (TIRBM) (combined)	58.7±0.5%
VQ (*K*=1,600) (Coates & Ng, 2011a)	54.9±0.4%
SC (*K*=1,600) (Coates Ng, 2011a)	59.0±0.8%
deep NN (Coates Ng, 2011b)	60.1±1.0%

The object classification task on STL-10 dataset was also performed, which is more challenging due to the smaller number of labeled training examples (100 per class for each training fold). Since the original images are 96x96 pixels, we down-sampled the images into 32x32 pixels, while keeping the RGB channels. We followed the same unsupervised training and classification pipeline as we did for CIFAR-10. As reported in Table 13, there were consistent improvements in classification accuracy by incorporating the various transformations in learning algorithms. Finally, 58.7% accuracy was achieved using 1,600 filters, which is competitive to the best published single layer result (59.0%).

[00214] The object detection techniques described herein may be implemented by one or more computer programs executed by one or more processors. The computer programs include processor-executable instructions that are stored on a non-transitory tangible computer readable medium. The computer programs may also include stored data. Non-limiting examples of the non-transitory tangible computer readable medium are nonvolatile memory, magnetic storage, and optical storage.

[00215] Some portions of the above description present the techniques (including models and machines) described herein in terms of algorithms and symbolic representations of operations on information. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. These operations, while described functionally or logically, are understood to be implemented by computer programs.

Furthermore, it has also proven convenient at times to refer to these arrangements of operations as modules or by functional names, without loss of generality.

[00216] In an example application, the object detection techniques are applied to image data captured by an imaging device, such as a camera. In this application the visible units in the models and machines described above represent intensity values for pixels in an image. While specific reference is made to detecting and manipulating objects in image data, the concept described herein are also extendable to other types of computer vision problems.

[00217] Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[00218] Certain aspects of the described techniques include process steps and instructions described herein in the form of an algorithm. It should be noted that the described process steps and instructions could be embodied in software, firmware or hardware, and when embodied in software, could be downloaded to reside on and be operated from different platforms used by real time network operating systems.

[00219] The present disclosure also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored on a computer readable medium that can be accessed by the computer. Such a computer program may be stored in a tangible computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media

suitable for storing electronic instructions, and each coupled to a computer system bus. Furthermore, the computers referred to in the specification may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

[00220] The algorithms and operations presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatuses to perform the required method steps. The required structure for a variety of these systems will be apparent to those of skill in the art, along with equivalent variations. In addition, the present disclosure is not described with reference to any particular programming language. It is appreciated that a variety of programming languages may be used to implement the teachings of the present disclosure as described herein.

[00221] The foregoing description of the embodiments has been provided for purposes of illustration and description. It is not intended to be exhaustive or to limit the disclosure. Individual elements or features of a particular embodiment are generally not limited to that particular embodiment, but, where applicable, are interchangeable and can be used in a selected embodiment, even if not specifically shown or described. The same may also be varied in many ways. Such variations are not to be regarded as a departure from the disclosure, and all such modifications are intended to be included within the scope of the disclosure.

CLAIMS

What is claimed is:

1. A computer-implemented method for classifying objects in an image, comprising:

constructing a point-wise gated Boltzmann machine having a visible layer of units, a corresponding switching unit for each of the visible units and at least one hidden layer of units, where the visible units represent intensity values for pixels in an image and the switching units determine which hidden units generate corresponding visible units;

receiving data for an image captured by a imaging device; and

classifying objects in the image data using the point-wise gated Boltzmann machine, where the point-wise gated Boltzmann machine is implemented as computer-readable instructions executed by a computer processor.

2. The method of claim 1 wherein constructing the point-wise gated Boltzmann machine further comprises partitioning the hidden units into groups of hidden units, such that each group of hidden units defines a distinct distribution over the visible units.

3. The method of claim 1 further comprises training the point-wise gated Boltzmann machine with stochastic gradient descent using contrastive divergence.

4. The method of claim 3 further comprises applying alternating Gibbs sampling for approximate inference.

5. The method of claim 1 further comprises training the point-wise gated Boltzmann machine using labeled training data.

6. The method of claim 5 further comprises training by connecting label units to a subset of hidden units, where label units are indicative of an object to be classified.

7. The method of claim 6 further comprises partitioning the hidden units into groups of hidden units, where select groups of hidden unit indicative of objects to be classified, and connecting label units only to hidden units in the select group of hidden units.

8. The method of claim 1 wherein a switching unit generates a binary output having a value of one only when its corresponding visible unit is assigned to the r-th components and its conditional probability given hidden units follows a multinomial distribution over R categories.

9. The method of claim 1 wherein an energy function of the point-wise gated Boltzmann machine is defined as:

$$E^{M}(\mathbf{v},\mathbf{z},\mathbf{h}) = -\sum_{r=1}^{R}\sum_{i=1}^{D}\sum_{k=1}^{K_r}(z_i^r v_i)W_{ik}^r h_k^r \qquad (1)$$
$$-\sum_{r=1}^{R}\sum_{k=1}^{K_r} b_k^r h_k^r - \sum_{r=1}^{R}\sum_{i=1}^{D}(z_i^r v_i)c_i^r,$$
$$\text{s.t.} \quad \sum_{r=1}^{R} z_i^r = 1,\ i = 1,\cdots,D.$$

where R mixture component has a multinomial switch unit denoted $z_i \in \{1,\ldots, R\}$ for each visible unit v_i, v, z^r and h are the visible, switch and hidden unit binary vectors, respectively, and the model parameters W_{ik}^r, b_k^r, c_i^r are the weights, hidden biases, and the visible biases of r-th component.

10. The method of claim 9 wherein the conditional probabilities are defined as:

$$P(h_k^r = 1 \mid \mathbf{z},\mathbf{v}) = \sigma\left((\mathbf{z}^r \odot \mathbf{v})^T \mathbf{W}_{:k}^r + b_k^r\right), \qquad (2)$$

$$P(v_i = 1 \mid \mathbf{z},\mathbf{h}) = \sigma\left(\sum_r z_i^r (\mathbf{W}_{i:}^r \mathbf{h}^r + c_i^r)\right), \qquad (3)$$

$$P(z_i^r = 1 \mid \mathbf{v},\mathbf{h}) = \frac{\exp(v_i(\mathbf{W}_{i:}^r \mathbf{h}^r + c_i^r))}{\sum_s \exp(v_i(\mathbf{W}_{i:}^s \mathbf{h}^s + c_i^s))}, \qquad (4)$$

where we use W_i^r, to denote i-th row, and $W_{:k}^r$ to denote k-th column of the matrix W^r.

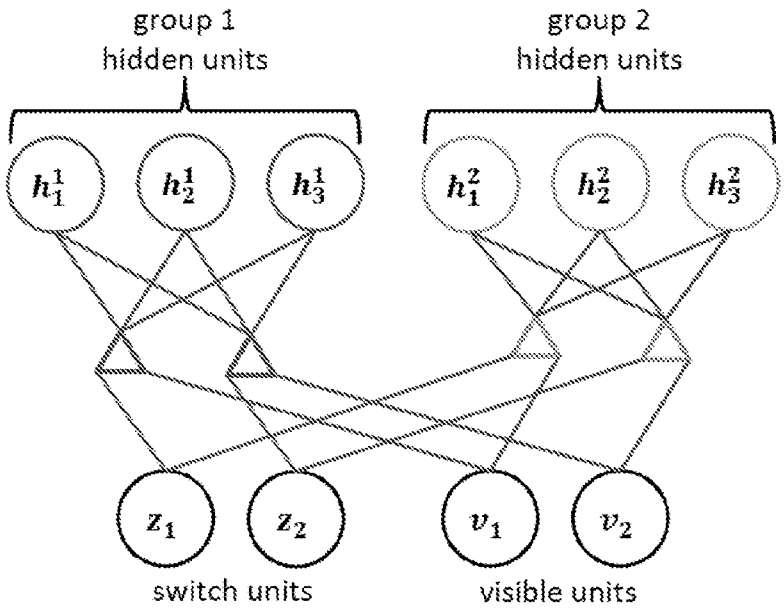

group 1
hidden units
group 2
hidden units
h_1^1
h_2^1
h_3^1
h_1^2
h_2^2
h_3^2
z_1
z_2
v_1
v_2
switch units
visible units

FIG. 1A

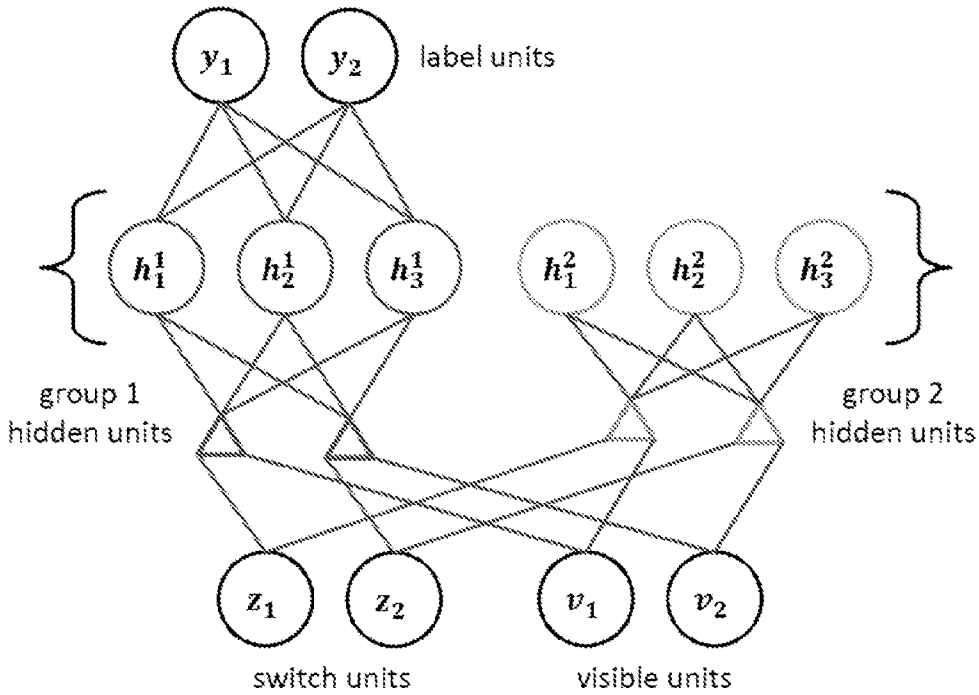

y_1
y_2
label units
h_1^1
h_2^1
h_3^1
h_1^2
h_2^2
h_3^2
group 1
hidden units
group 2
hidden units
z_1
z_2
v_1
v_2
switch units
visible units

FIG. 1B

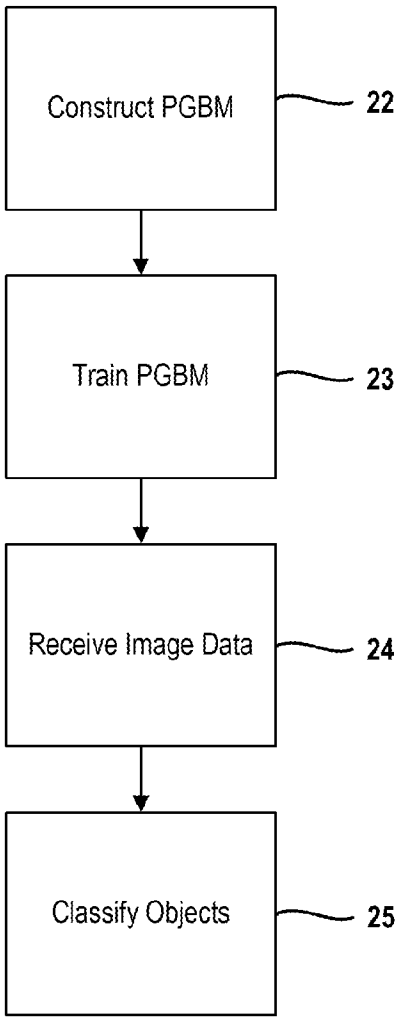

Construct PGBM
22
Train PGBM
23
Receive Image Data
24
Classify Objects
25

FIG. 2

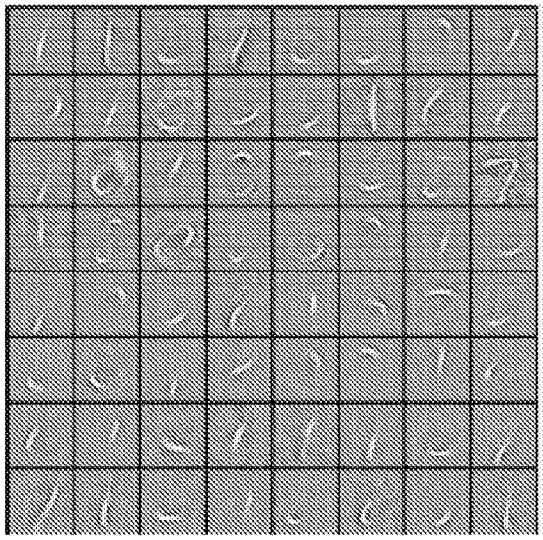


FIG. 3A

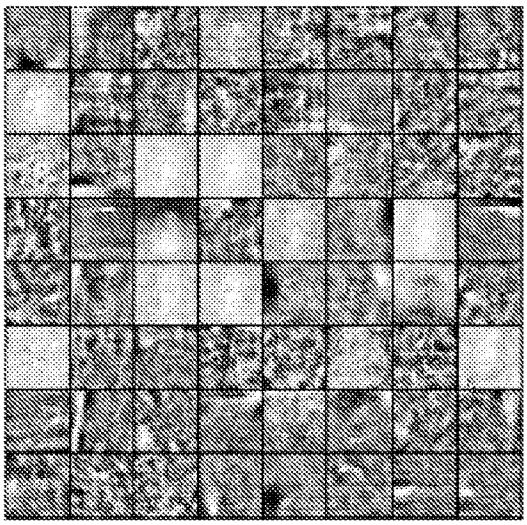


FIG. 3B

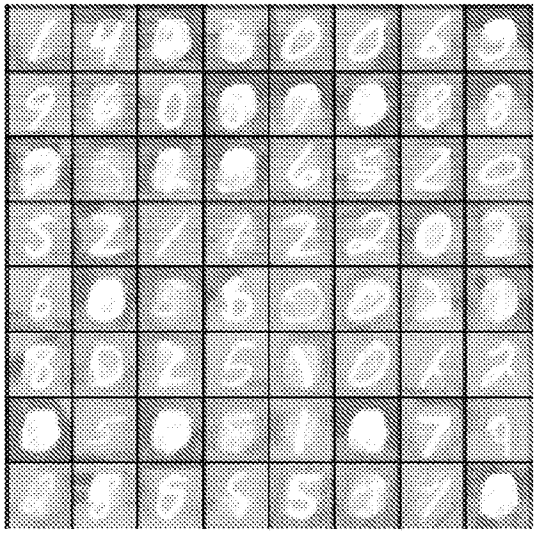


FIG. 3C

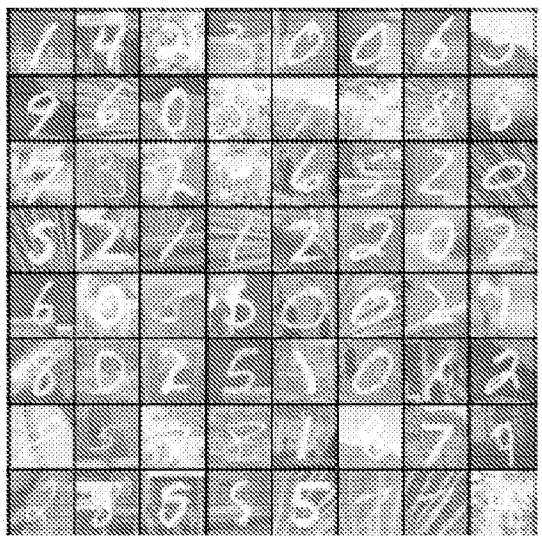


FIG. 3D

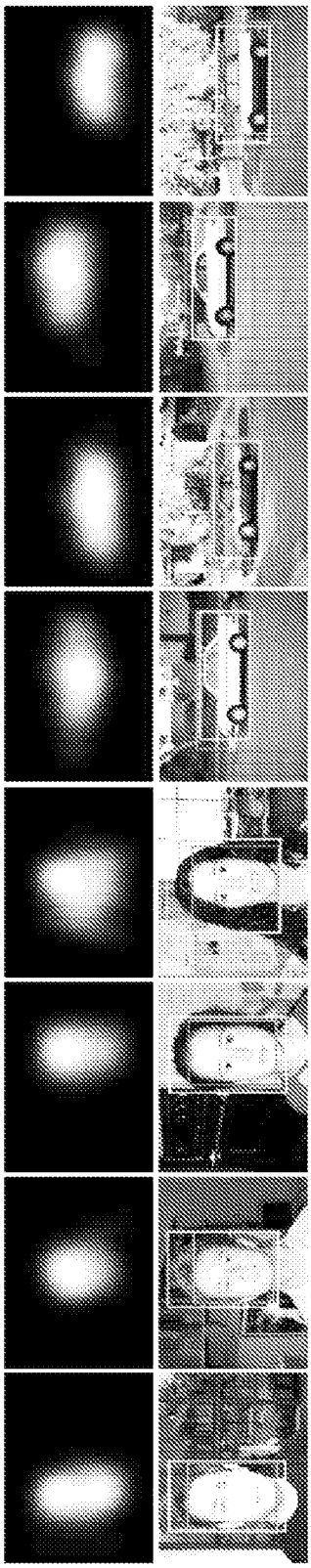


FIG. 4

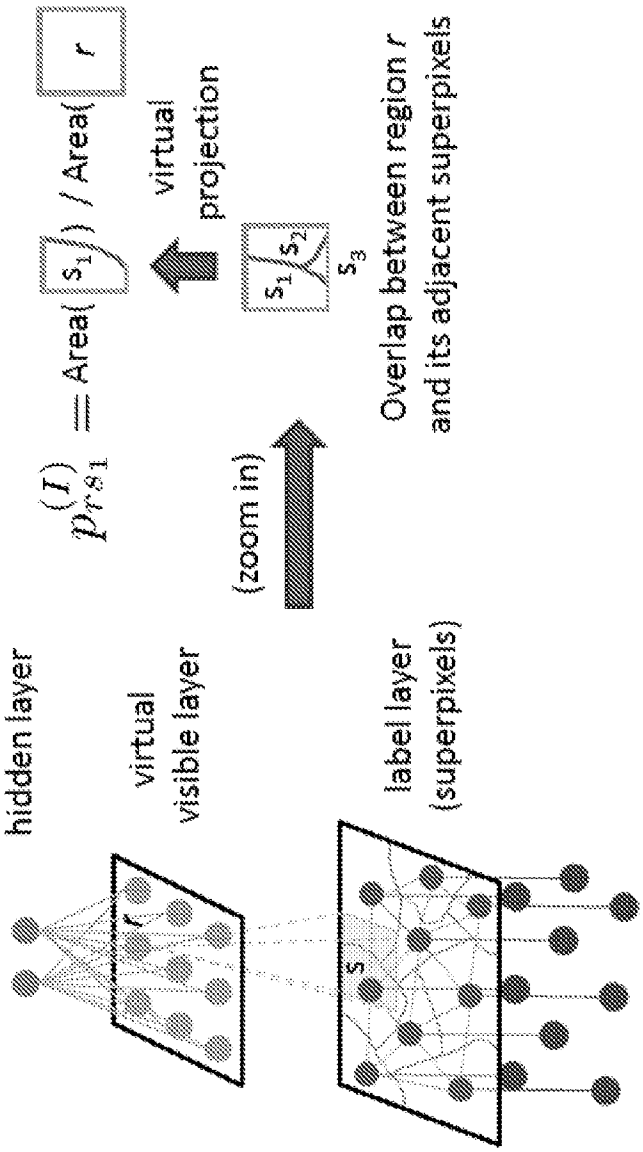

hidden layer
virtual visible layer
label layer (superpixels)
(zoom in)
virtual projection
Overlap between region r and its adjacent superpixels
$p_{rs_1}^{(I)} = \text{Area}(s_1) / \text{Area}(r)$

FIG. 5

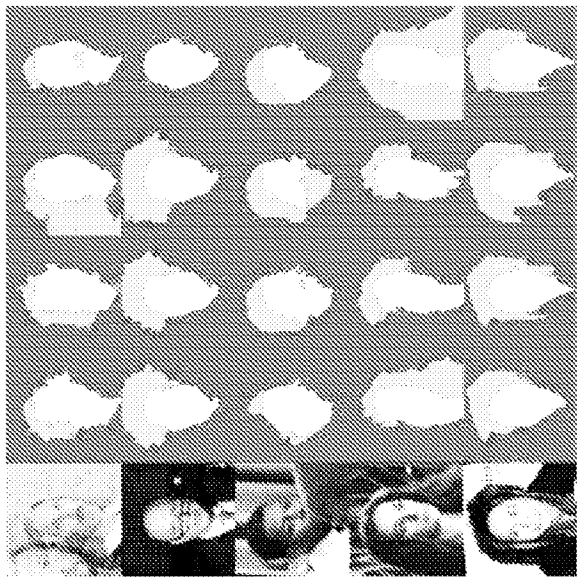


FIG. 6B

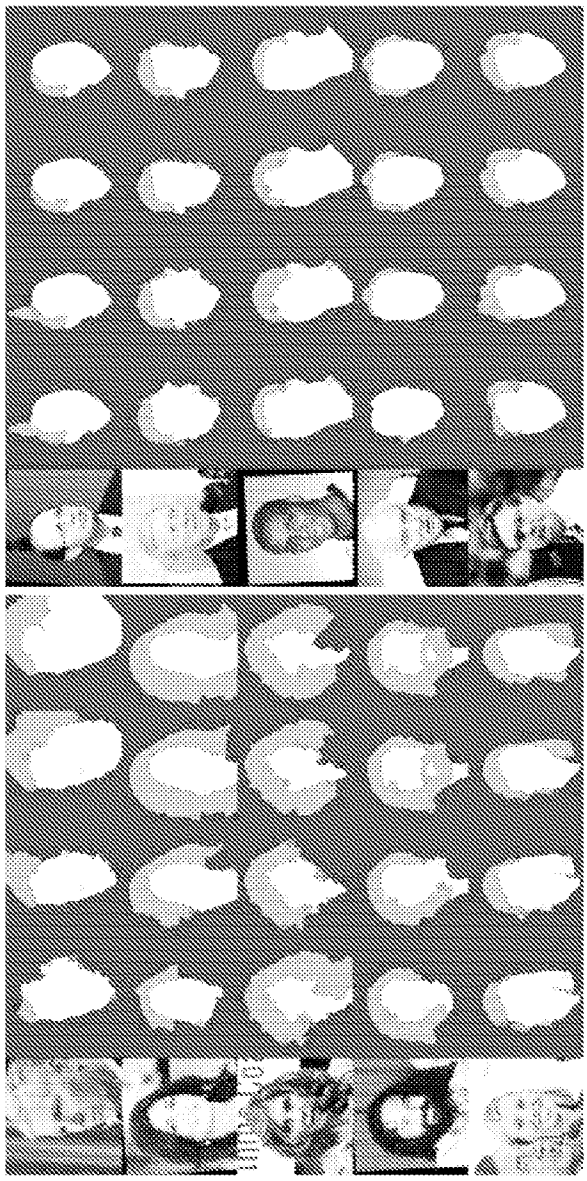


FIG. 6A

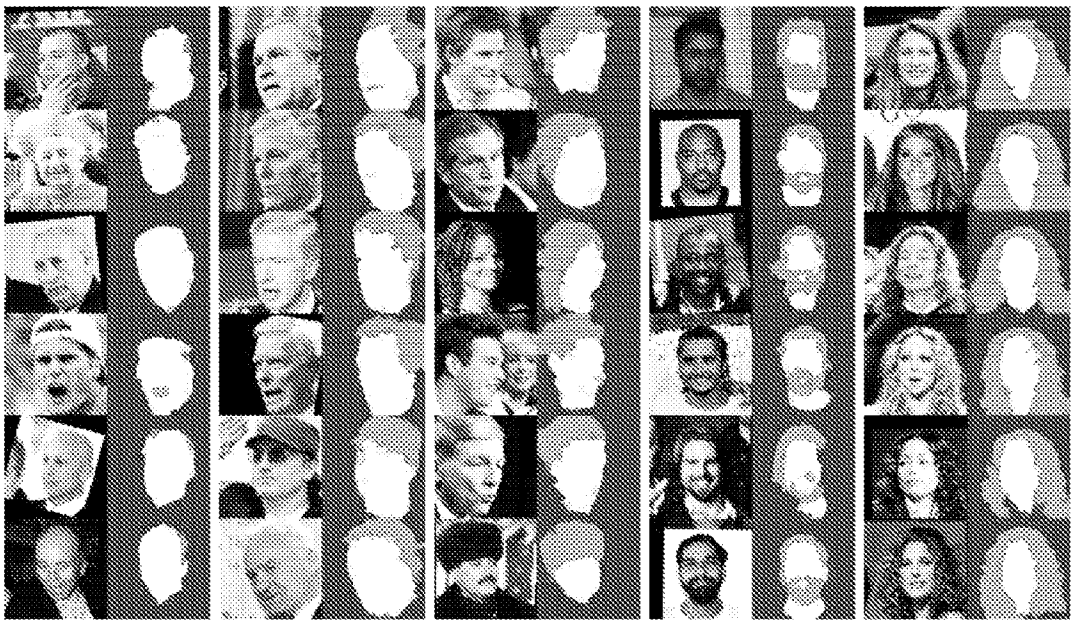


FIG. 7

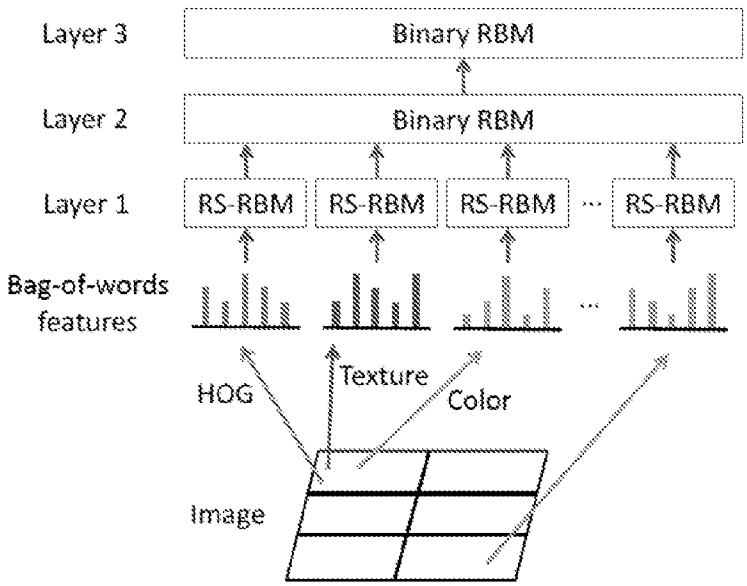

Layer 3
Binary RBM
Layer 2
Binary RBM
Layer 1
RS-RBM
RS-RBM
RS-RBM
RS-RBM
Bag-of-words features
HOG
Texture
Color
Image

FIG. 8

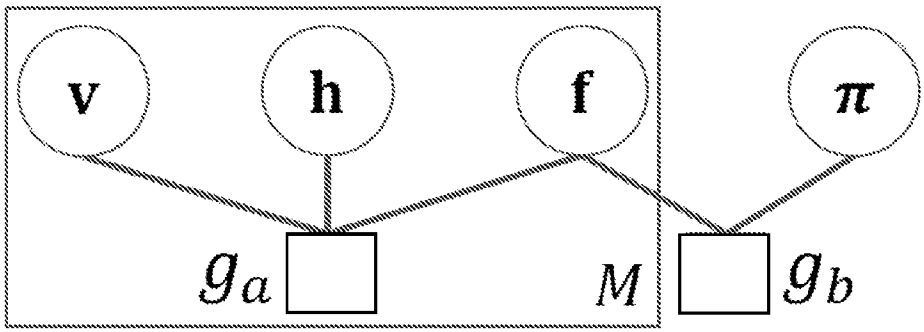


FIG. 9A

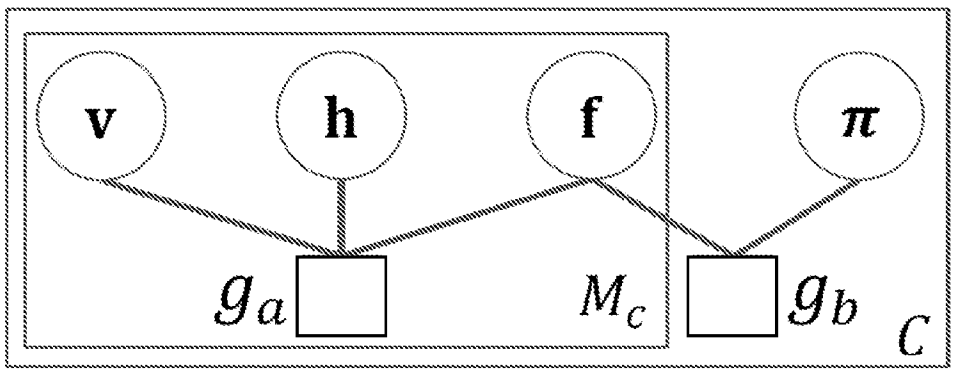


FIG. 9B

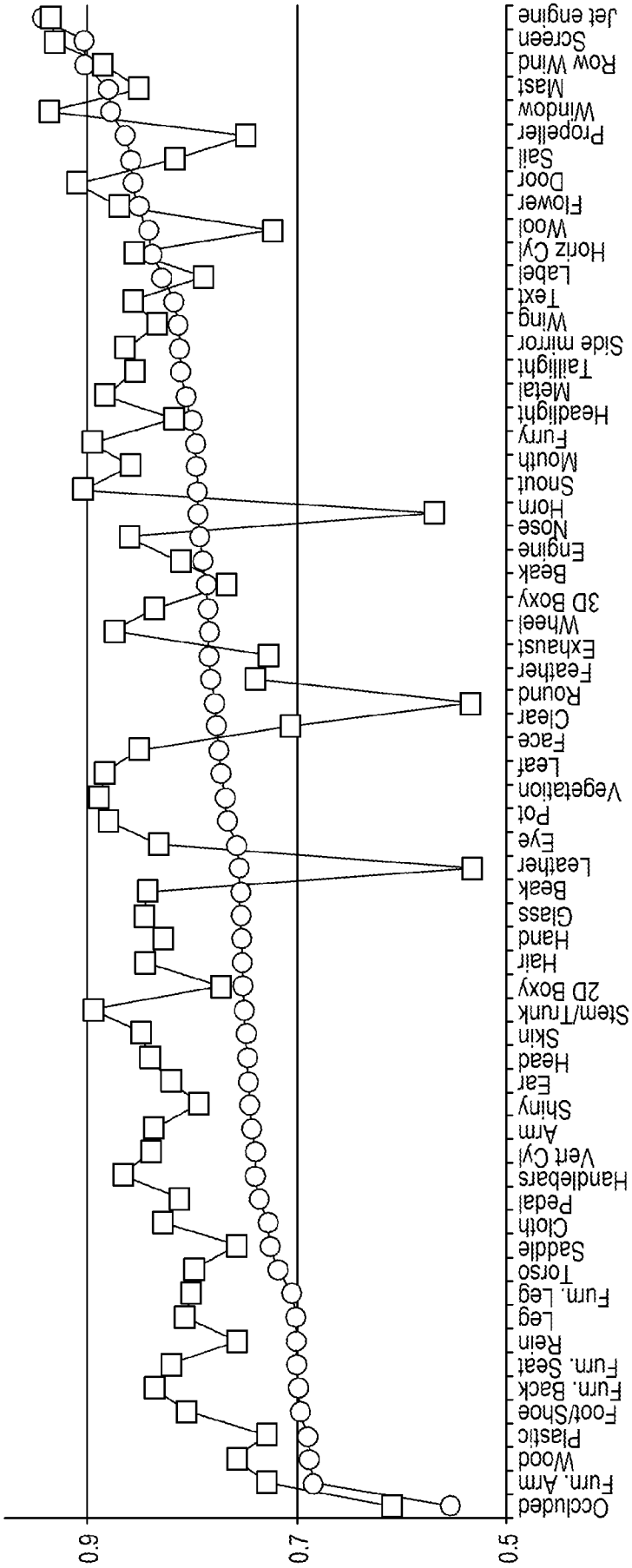

0.9
0.7
0.5
Occluded
Furn. Arm
Wood
Plastic
Foot/Shoe
Furn. Back
Furn. Seat
Rein
Leg
Furn. Leg
Torso
Saddle
Cloth
Pedal
Handlebars
Vert Cyl
Arm
Shiny
Ear
Head
Skin
Stem/Trunk
2D Boxy
Hair
Hand
Glass
Beak
Leather
Eye
Pot
Vegetation
Leaf
Face
Clear
Round
Feather
Exhaust
Wheel
3D Boxy
Beak
Engine
Nose
Horn
Snout
Mouth
Furry
Headlight
Metal
Taillight
Side mirror
Wing
Text
Label
Horiz Cyl
Wool
Flower
Door
Sail
Propeller
Window
Mast
Row Wind
Screen
Jet engine

FIG. 10

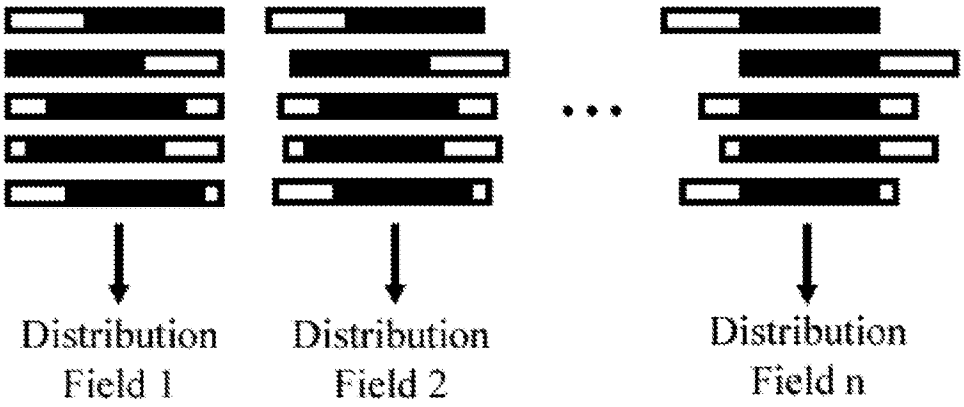

Distribution Field 1
Distribution Field 2
Distribution Field n

FIG. 11

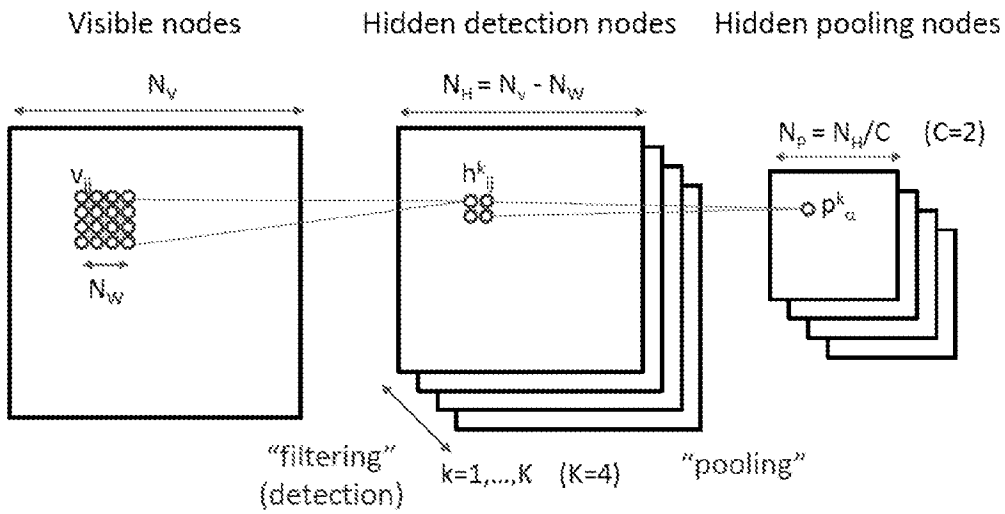

Visible nodes
Hidden detection nodes
Hidden pooling nodes
N_V
$N_H = N_V - N_W$
$N_P = N_H/C$ (C=2)
v_{ij}
h^k_{ij}
p^k_α
N_W
"filtering" (detection)
k=1,...,K (K=4)
"pooling"

FIG. 12

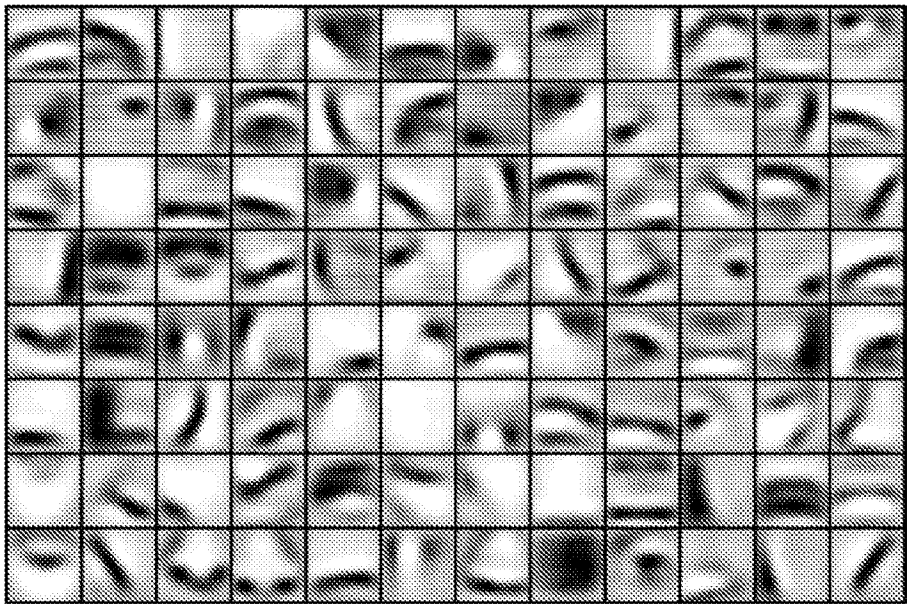


FIG. 13A

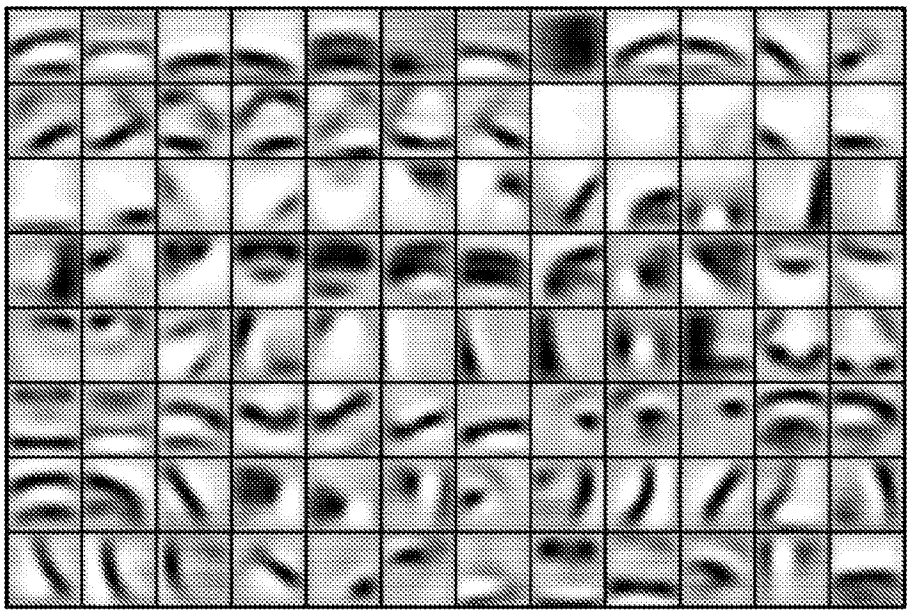


FIG. 13B


LFW-a
deep l2
deep l1
SIFT
original
LFW-a
deep l2
deep l1
SIFT
original

FIG. 14

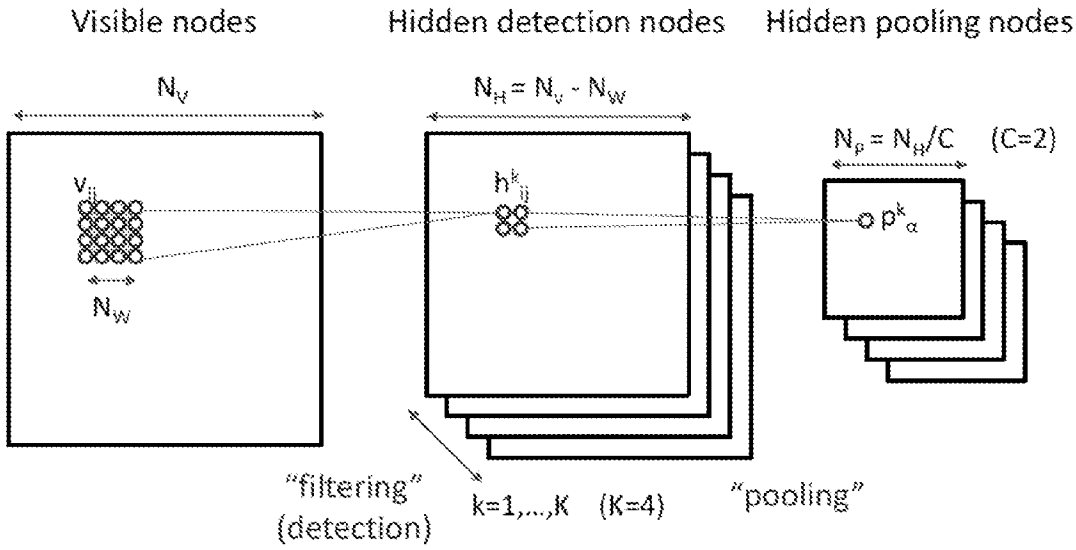

Visible nodes
Hidden detection nodes
Hidden pooling nodes
N_V
$N_H = N_V - N_W$
$N_P = N_H/C$ (C=2)
v_{ij}
h^k_{ij}
p^k_α
N_W
"filtering" (detection)
k=1,...,K (K=4)
"pooling"

FIG. 15

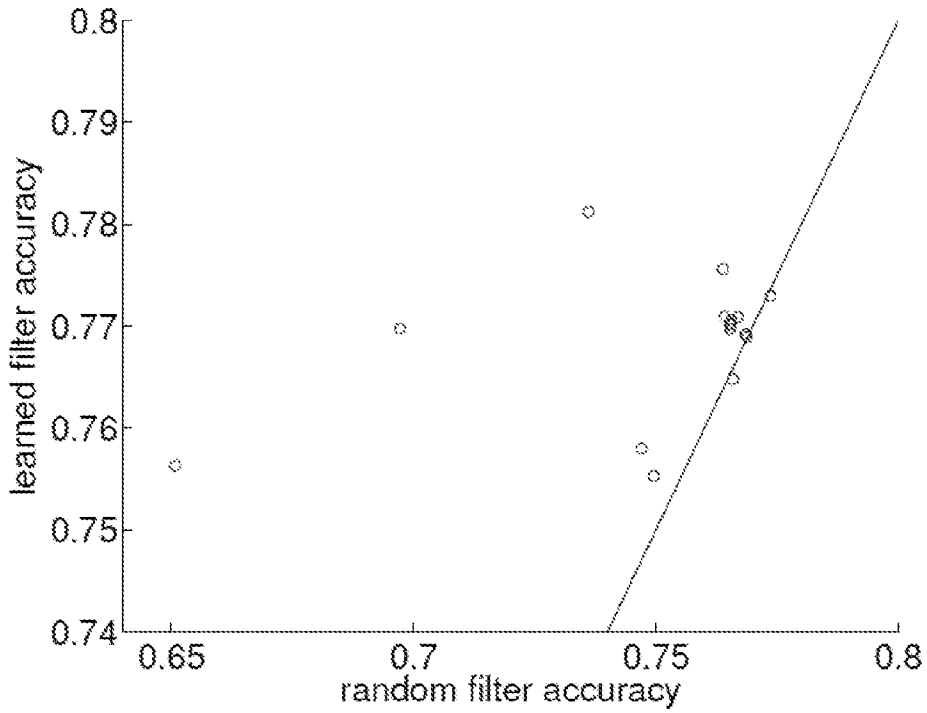

0.8
0.79
0.78
0.77
0.76
0.75
0.74
learned filter accuracy
0.65
0.7
0.75
0.8
random filter accuracy

FIG. 16

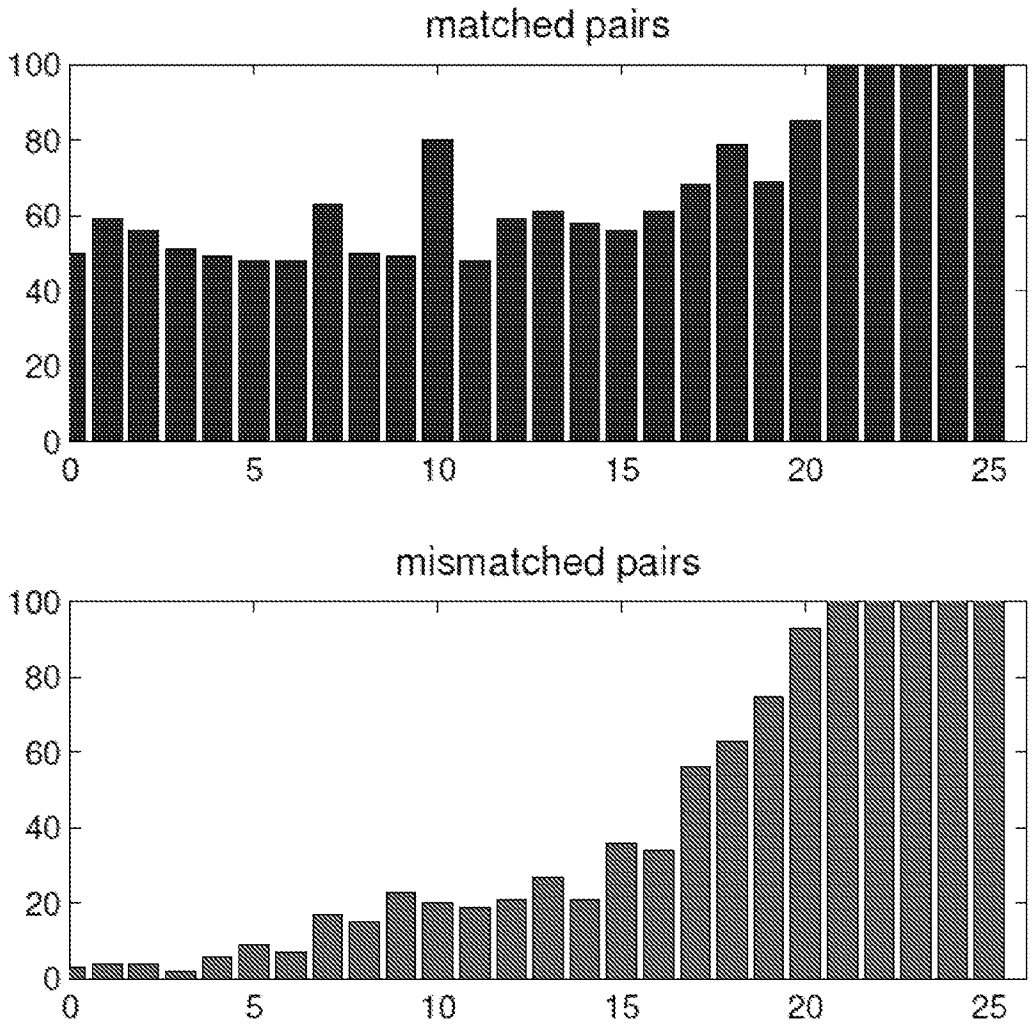

matched pairs
100
80
60
40
20
0
0
5
10
15
20
25
mismatched pairs
100
80
60
40
20
0
0
5
10
15
20
25

FIG. 17

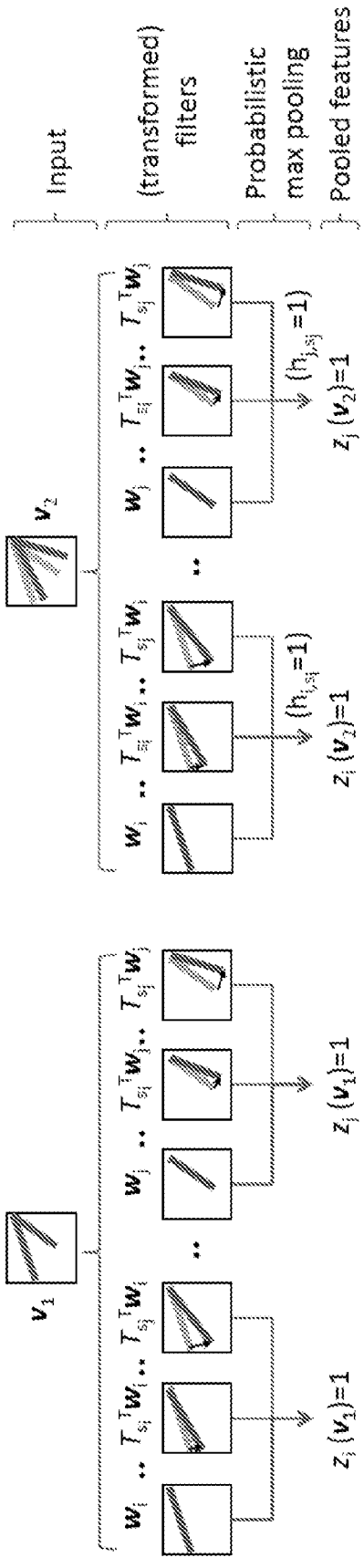

Input
(transformed) filters
Probabilistic max pooling
Pooled features
v_1
v_2
w_i .. $T_{s_i}^T w_i$ .. $T_{s_j}^T w_i$
w_j .. $T_{s_i}^T w_j$ .. $T_{s_j}^T w_j$
$z_i(v_1)=1$
$z_i(v_2)=1$
$(h_{i,s_i}=1)$
$(h_{i,s_j}=1)$

FIG. 18

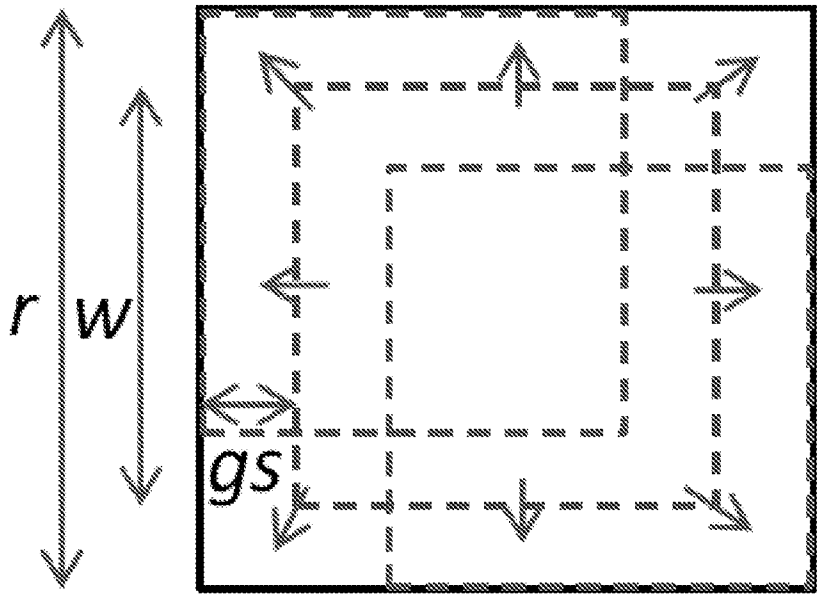


FIG. 19A

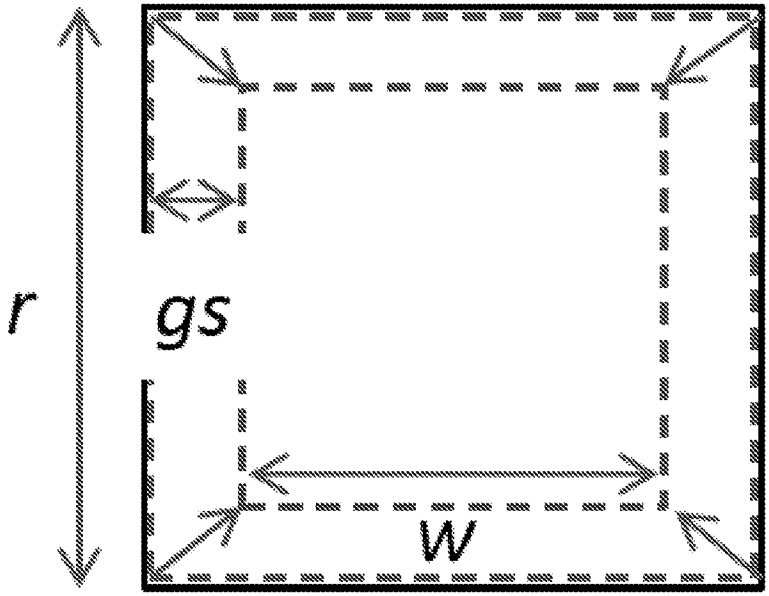


FIG. 19B

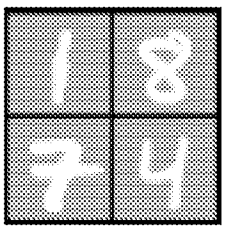


FIG. 20A

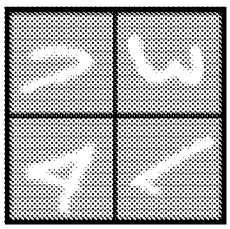


FIG. 20B

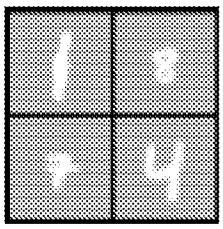


FIG. 20C

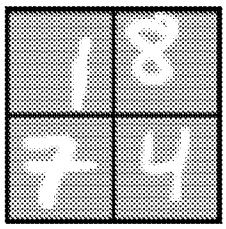


FIG. 20D

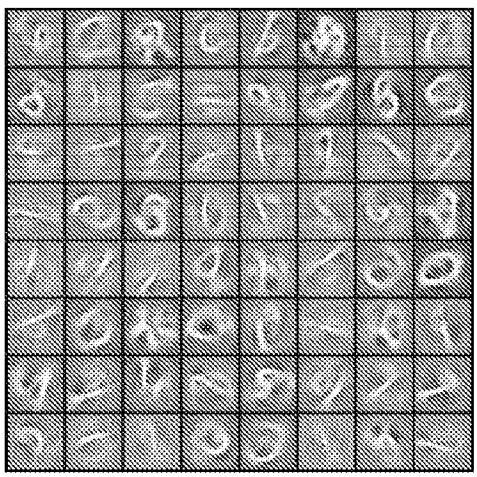


FIG. 20E

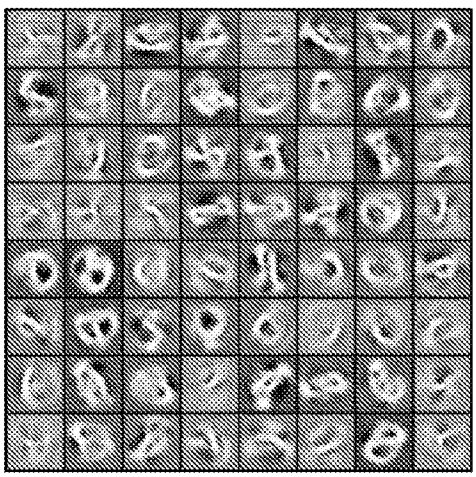


FIG. 20F

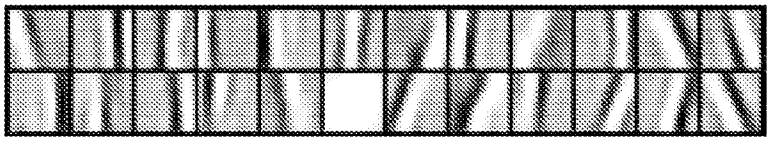


FIG. 21A

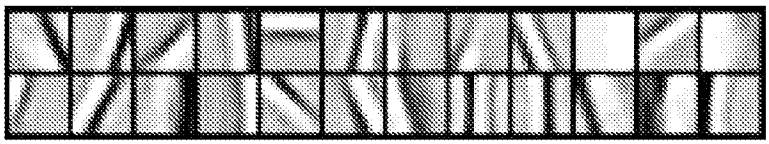


FIG. 21B

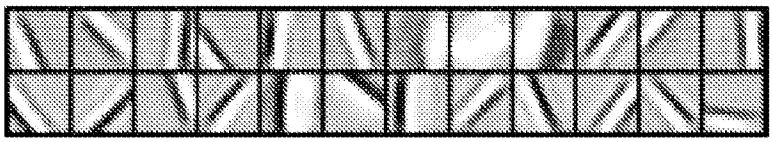


FIG. 21C

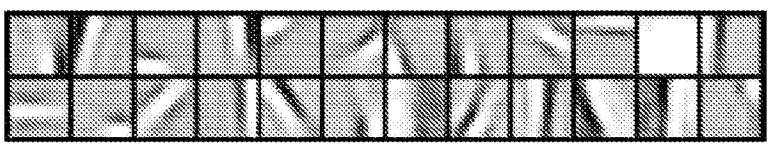


FIG. 21D

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/043206

A. CLASSIFICATION OF SUBJECT MATTER

G06T 7/20(2006.01)i, G06K 9/00(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)
G06T 7/20; G06K 9/00; G06K 9/62; G06T 7/00; G09G 5/00

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: point-wise gated Boltzmann machine, regonize, object, image

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KIHYUK SOHN et al., `Learning and Selecting Features Jointly with Point-wise Gated Boltzmann Machines`, Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA, 07 April 2013 (http://jmlr.org/proceedings/papers/v28/sohn13.pdf) See pages 2-5.	1-10
A	EP 0499627 B1 (Sarnoff Corporation) 20 May 1998 See abstract, page02, line 02 - page03, line 03, claims 1 - 3.	1-10
A	WO 2012-091276 A1 (INDUSTRY FOUNDATION OF CHONNAM NATIONAL UNIVERSITY et al.) 05 July 2012 See abstract, paragraph[26] - paragraph[48], claims 1 - 6.	1-10
A	US 2010-0183195 A1 (SHARMA VINAY) 22 July 2010 See abstract, paragraph[19] - paragraph[63], claim 1.	1-10
A	US 2009-0102858 A1 (EGGERS HELMUTH et al.) 23 April 2009 See abstract, paragraph[16] - paragraph[44], claims 1 - 24.	1-10

☐ 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
"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
30 September 2014 (30.09.2014)	**02 October 2014 (02.10.2014)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
PARK, Sang Cheol
Telephone No. +82-42-481-8372

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2014/043206

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0499627 B1	20/05/1998	DE 69032326 D1	25/06/1998
		DE 69032326 T2	10/09/1998
		EP 0499627 A1	26/08/1992
		EP 0499627 A1	24/09/1997
		JP 03178719 B2	25/06/2001
		US 05063603 A	05/11/1991
		WO 91-06921 A1	16/05/1991
WO 2012-091276 A1	05/07/2012	KR 10-1195939 B1	29/10/2012
		KR 10-2012-0074591 A	06/07/2012
US 2010-0183195 A1	22/07/2010	US 8340354 B2	25/12/2012
US 2009-0102858 A1	23/04/2009	DE 102006047777 A1	20/09/2007
		EP 1997093 A1	03/12/2008
		EP 1997093 B1	13/07/2011
		JP 05121737 B2	16/01/2013
		JP 2009-530695 A	27/08/2009
		WO 2007-107259 A1	27/09/2007