Closed-Loop Deep Vision

Gustavo Carneiro, Zhibin Liao, Tat-Jun Chin
Australian Centre for Visual Technologies
University of Adelaide

***Abstract*—There has been a resurgence of interest in one of the most fundamental aspects of computer vision, which is related to the existence of a feedback mechanism in the inference of a visual classification process. Indeed, this mechanism was present in the first computer vision methodologies, but technical and theoretical issues imposed major roadblocks that forced researchers to seek alternative approaches based on pure feed-forward inference. These open loop approaches process the input image sequentially with increasingly more complex analysis steps, and any mistake made by intermediate steps impair all subsequent analysis tasks. On the other hand, closed-loop approaches involving feed-forward and feedback mechanisms can fix mistakes made during such intermediate stages. In this paper, we present a new closed-loop inference for computer vision problems based on an iterative analysis using deep belief networks (DBN). Specifically, an image is processed using a feed-forward mechanism that will produce a classification result, which is then used to sample an image from the current belief state of the DBN. Then the difference between the input image and the sampled image is fed back to the DBN for re-classification, and this process iterates until convergence. We show that our closed-loop vision inference improves the classification results compared to pure feed-forward mechanisms on the MNIST handwritten digit dataset [1] and the Multiple Object Categories [2] containing shapes of horses, dragonflies, llamas and rhinos.**

I. Introduction

Some of the most famous computer vision scientists have defended closed-loop image analysis processes [3], [4], [5], [6], [7], [8], [9], [10], [11] for decades, but the following roadblocks were found during these initial developments [7]: 1) lack of computational resources to process images, and 2) large representational gap between the features extracted from the image and the abstract 3-D model features that were used to represent the visual classes. These obstacles encouraged the development of simpler open loop mechanisms that are currently dominant in the field [12], [13], [14], [15].

Open loop mechanisms [16] take an image and process it in several steps, where each step needs a model that is built using an annotated training set of images containing examples of the analysis output. For example, the first step can estimate the topic structure, the second step can produce a list of annotations using semantic keywords, and so forth. The success of open loop approaches stems from: 1) the maturity of machine learning techniques, which can now produce quite robust models; 2) the size and richness of training databases available to estimate the parameters of these models; and 3) the computational resources available for handling the training and inference procedures of such models. One of the main issues with these open loop approaches is that the more complex

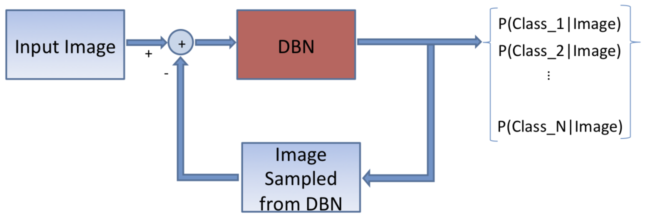

Fig. 1. Closed-loop deep inference process.

analysis stages require large capacity models, which in turn need larger training sets. However, the size of training sets tends to shrink for the more complicated analysis tasks (e.g., 3-D modeling), which does not allow for a robust estimation of the model parameters. Researchers have addressed this issue by trying to increase the size of the training set for these complicated analysis stages via crowdsourcing [17], but there are at least three problems with this solution: 1) the acquisition of such training sets is intractable because of the combinatorial nature of the exponentially more complex image analysis stages; 2) the large complexity involved in the training and testing procedures of such models can impose a strain on the computational resources because of the high model capacity and the size of the training sets; and 3) the reliability of manual annotations is likely to decrease as the analysis stages become more challenging.

Some of these issues have already been noticed by a few researchers in the field [2], [18], who have proposed closed-loop inference mechanisms for image analysis. Therefore, it is foreseeable that closed-loop inference methods will receive increasingly more attention by the computer vision community. In general, closed-loop methods will alleviate the need to acquire these massive and unreliably annotated databases because of their ability to automatically fix analysis mistakes made during the iterative inference process. This ability to fix analysis mistakes will also allow for models with relatively smaller capacity than the current ones used in open loop approaches.

In this paper, we explore deep belief networks (DBN) for the implementation of our closed-loop inference approach. The major advantage of designing a closed-loop inference using DBN lies in its ability to process images using feed-forward and feedback mechanisms [2]. Our main contribution with this paper is the development of an inference mechanism that explores a feed-forward [19] model that estimates the posterior probability of a visual class given the input image (see Fig. 1), which in turn uses this posterior distribution to generate [20] an image given the DBN belief state. Then the difference between the original input image and the generated image will be fed back to the DBN in an iterative process that runs until convergence. We show empirically that the posterior distribution obtained after the convergence of this iterative process tends to fix a relatively large proportion of the mistakes committed by a pure open loop inference. Specifically, we show with the MNIST handwritten dataset [1] that for several DBN structures we obtain relative improvements in the order of 5% to 10%. Furthermore, we obtain significantly better recognition results on the Multiple Object Categories database [2] compared both with the open loop inference and the original shape Boltzmann machine proposed by Eslami et al. [2].

A. Literature Review

The issue of open versus closed-loop inference is longstanding in the field of computer vision. As pointed out by Hoiem et al. [18], Marr [16] proposed an open loop system that consisted of a sequence of increasingly more complex tasks, where any mistake made by some of the initial steps is propagated to subsequent stages. On the other hand, Barrow and Tenenbaum [3] proposed a closed-loop inference, where the analysis processes collaborated with each other in order to fix eventual mistakes made by each stage. Indeed, most of the seminal papers in the field [3], [4], [5], [6], [9], [10] used a verification stage that relied on a feedback mechanism, which enabled changes in scene interpretation. More recently, the top-down bottom-up inference of Zhu and Mumford [11] also presents a closed-loop inference that allows for updates in the analysis process. Nevertheless, all these papers rely on the extraction of features from the image, and the closed-loop analysis process can vary the scene interpretation using these image features as input, which means that these methodologies can only reconstruct these input features instead of reconstructing the actual input image, as we propose in this paper.

One the main inspirations for this paper is the work by Hoiem et al. [18], who use the concept of intrinsic images to represent maps that describe specific aspects of the image analysis. For instance, Hoiem et al. designed systems to produce surface maps, occlusion maps, depth maps and object maps. Then in their closed-loop inference each one of those systems will interact with one another in order to constrain subsequent analysis. In comparison to that approach, our methodology offers a simpler unified mechanism that uses a feedback mechanism that produces an image, representing the current belief state of the DBN. Then the difference between the current belief state and the input image drives our closed-loop analysis process. Another related approach recently proposed is the shape Boltzmann Machine [2], which comprises a deep belief network that is able to learn shape models and to generate shapes given partial image data information. Eslami et al. [2] also work on a classification problem, but they have not explored the generative capability of the DBN in order to implement a closed-loop inference mechanism, as we do in this paper.

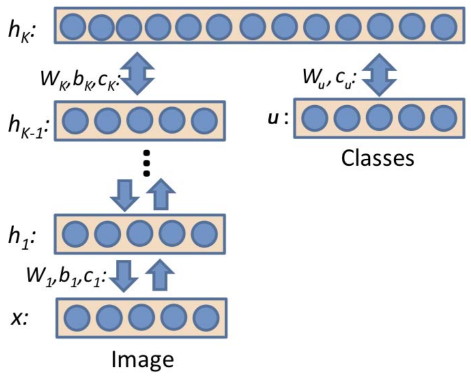

Fig. 2. Deep belief network.

II. Methodology

In this section we first introduce the deep belief network used in this work and the training approach. We then explain the feedback and feed-forward mechanisms, and we conclude the section with the algorithm used for the closed-loop inference.

A. Deep Belief Network

The DBN depicted in Fig. 2 represents the core structure of the networks used in this paper, which use as input an image I with size $M \times M$, represented by the vector $\mathbf{x} \in [0,1]^{M \times M}$ in the bottom layer. This bottom layer is called a visible layer, while the hidden layers above it are represented by the nodes $\mathbf{h}_k \in [0,1]^{\#\text{nodes}_k}$ for layers $k \in \{1,..,K\}$ and $\#\text{nodes}_k$ denotes the number of nodes in layer $\mathbf{h}_k$. Finally, the nodes in layer $\mathbf{u} \in \{0,1\}^U$ (with $\|\mathbf{u}\|_1 = 1$) in the top visible layer represent the multi-class classification of the input $\mathbf{x}$.

The joint distribution of the DBN in Fig. 2 is defined as follows:

$$p(\{\mathbf{h}_k\}_{k=1}^K, \mathbf{x}, \mathbf{u}|\Theta) = q(\mathbf{h}_1|\mathbf{x},\Theta)\left[\prod_{k=2}^{K-1} q(\mathbf{h}_k|\mathbf{h}_{k-1},\Theta)\right] p(\mathbf{h}_K, \mathbf{h}_{K-1}, \mathbf{u}|\Theta), \tag{1}$$

$$= p(\mathbf{x}|\mathbf{h}_1,\Theta)\left[\prod_{k=2}^{K-1} p(\mathbf{h}_{k-1}|\mathbf{h}_k,\Theta)\right] p(\mathbf{h}_K, \mathbf{h}_{K-1}, \mathbf{u}|\Theta), \tag{2}$$

where the DBN parameters are represented by $\Theta = \{\mathbf{W}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i \in \{1,\ldots,K\}} \bigcup \{\mathbf{W}_u, \mathbf{c}_u\}$, and $p(.)$ and $q(.)$ represent probability functions defined as:

$$\begin{aligned} p(\mathbf{x}(j)=1|\mathbf{h}_1,\Theta) &= \sigma(\mathbf{c}_1(j)+\mathbf{W}_1(:,j)^\top\mathbf{h}_1) \\ p(\mathbf{h}_{k-1}(j)=1|\mathbf{h}_k,\Theta) &= \sigma(\mathbf{c}_k(j)+\mathbf{W}_k(:,j)^\top\mathbf{h}_k) \\ q(\mathbf{h}_1(i)=1|\mathbf{x},\Theta) &= \sigma(\mathbf{b}_1(i)+\mathbf{W}_1(i,:)\mathbf{x}) \\ q(\mathbf{h}_k(i)=1|\mathbf{h}_{k-1},\Theta) &= \sigma(\mathbf{b}_k(i)+\mathbf{W}_k(i,:)\mathbf{h}_{k-1}), \end{aligned} \tag{3}$$

$$p(\mathbf{h}_K,\mathbf{h}_{K-1},\mathbf{u}|\Theta) = \frac{1}{Z}\exp(-\mathbf{E}), \tag{4}$$

where $\mathbf{E} = \mathbf{c_K^\top h_{K-1}} + \mathbf{c_u^\top u} + \mathbf{b_K^\top h_K} + \mathbf{h_K^\top W_h h_{K-1}} + \mathbf{h_K^\top W_u u}$, with $\mathbf{x}(j)$ denoting the j^{th} element of the vector $\mathbf{x}$ (and similarly for the other vectors in (3)), $\mathbf{W}_k(:,j)$ representing the j^{th} column of matrix $\mathbf{W}_k$, $\mathbf{W}(j,:)$ denoting the j^{th} row, Z representing the partition factor, and $\sigma(y) = \frac{1}{1+\exp(-y)}$.

Notice that the same model naturally permits a feed-forward analysis (1) and a feedback analysis (2) of an image. In the sections below, we explain both the feed-forward and feedback mechanisms and how they can be used in order to build a closed-loop inference.

For the estimation of the model parameters, assume the existence of a set of training images $\mathcal{X} = \{(\mathbf{x},y)_n\}_{n=1}^{|\mathcal{X}|}$ with $y \in \{1,...,U\}$ denoting the class of image $\mathbf{x}_i$, where $|.|$ denotes set cardinality. For the learning process of the DBN parameters Θ, we assume that each pair of network layers is initially represented by a restricted Boltzmann machine (RBM) [21]. An RBM is a stochastic neural network comprising a complete bipartite graph separating a visible from a hidden layer. The maximum likelihood estimation of the RBM parameters $\mathbf{W}_k, \mathbf{c}_k, \mathbf{b}_k$ (for layer k) is based on contrastive divergence [21]. The pseudo-code for the training of the DBNs used in this paper is shown in Alg. 1.

Algorithm 1 DBN training for $\Theta^* = \arg\min_\Theta E(\mathcal{X}|\Theta)$

```
1: Set $\mathcal{V}^{(0)} = \{\mathbf{v}_n^{(0)}\}_{n=1}^{|\mathcal{X}|} = \mathcal{X}$ (i.e., $\mathbf{v}_n^{(0)} = \mathbf{x}_n$ for $n \in \{1,...,|\mathcal{X}|\}$)
2: for k = 1:K-1 do
3:     Estimate the RBM parameters $\mathbf{W}_k^*, \mathbf{b}_k^*, \mathbf{c}_k^*$ for layer $k$ with training set $\mathcal{V}^{(k-1)}$ using contrastive divergence [21].
4:     Build a new training set $\mathcal{V}^{(k)}$ with elements $\mathbf{v}^{(k)} = q(\mathbf{h} = \mathbf{1}|\mathbf{v}^{(k-1)}, \mathbf{W}_k^*, \mathbf{b}_k^*, \mathbf{c}_k^*)$
5: end for
6: Estimate the RBM parameters $\mathbf{W}_K^*, \mathbf{b}_K^*, \mathbf{c}_K^*, \mathbf{W}_u^*, \mathbf{c}_u^*$ for layer $K$ with training set $\mathcal{V}^{(k-1)}$, where each of its elements is associated with the $U-$dimensional binary vector $\{\mathbf{u}_i\}_{i=1}^{|\mathcal{X}|}$ (note that each $\mathbf{u}_i$ is a vector with zeros except at the $y_i^{th}$ dimension). The estimation of these parameters is also done with contrastive divergence.
```

B. Feed-forward and Feedback Mechanisms

The **feed-forward** inference is realized with (1) by taking the image $\mathbf{x}$, computing $q(\mathbf{h}_1(i)=1|\mathbf{x},\Theta)$ for all nodes i on the first layer and calculating $q(\mathbf{h}_k(i)=1|\mathbf{h}_{k-1},\Theta)$ for all nodes from layers 2 to $K-1$ in (3). Then, in order to classify the test image, we compute the free energy given by the activation of each class, as follows [19]:

$$\begin{aligned} f(\mathbf{x},\mathbf{u}) = &-\mathbf{c}_K^\top\mathbf{h}_{K-1} - \mathbf{c}_u^\top\mathbf{u} - \\ &\sum_i \log(1+\exp\{\mathbf{b}_i + \mathbf{W}_K(i,:)\mathbf{h}_{k-1} + \mathbf{W}_u(i,:)\mathbf{u}\}). \end{aligned} \tag{5}$$

The $\mathbf{u}$ vector that produces the lowest free energy is the most likely class label estimated from the feed-forward mechanism. Also, assuming that $\mathbf{u}_i \in \{0,1\}^U$ is a vector with zeros in all U dimensions except at the i^{th} position, we can build a vector of free energies, defined by: $[f(\mathbf{x},\mathbf{u}_1), f(\mathbf{x},\mathbf{u}_2), ..., f(\mathbf{x},\mathbf{u}_U)]$, which can be used to compute the posterior probability, as follows:

$$p(y=i|\mathbf{x}) = \frac{\exp\{-f(\mathbf{x},\mathbf{u}_i)\}}{\sum_{j=1}^U \exp\{-f(\mathbf{x},\mathbf{u}_j)\}}. \tag{6}$$

The **feedback** method uses (2) starting with a fixed value for $\mathbf{u}$. Then we use alternating Gibbs sampling to obtain an equilibrium sample from $p(\mathbf{h}_K,\mathbf{h}_{K-1},\mathbf{u}|\Theta)$, where $\mathbf{u}$ is fixed as explained above. This is followed by the computation of $p(\mathbf{h}_{k-1}|\mathbf{h}_k,\Theta)$ until we reach the bottom layer, where we again compute the probability distribution for $p(\mathbf{x}|\mathbf{h}_1,\Theta)$ and sample from it in order to obtain the generated image.

C. Closed-Loop Inference

This is the major contribution of this paper, and the idea is to perform several feed-forward and feedback steps by iterating prediction, data association and correction stages, similarly to the procedures present in visual tracking algorithms [22]. Hence, we approximate the closed-loop inference mechanism with a sequential inference model consisting of T iterations, and producing the following distribution of classes given the images generated at each iteration:

$$\begin{aligned} p(y^{(t)}|\{\mathbf{x}^{(l)}\}_{l=1}^t) \propto\, &(1/Z)p(\mathbf{x}^{(t)}|y^{(t)}) \\ &\sum_{j=1}^U p(y^{(t)}|y^{(t-1)}=j)p(y^{(t-1)}=j|\{\mathbf{x}^{(l)}\}_{l=1}^{t-1}), \end{aligned} \tag{7}$$

where the superscript (t) indicates the iteration index, and Z is a normalization factor. The gist of our approach consists of, at each iteration step, predicting the posterior distribution by combining the results from the previous iteration and a transition model that accounts for how unsure the DBN is about the classification of a specific class. Using this predictive distribution, the system generates several images and picks the one that maximizes the data association probability. Finally, by taking a difference between the input image $\mathbf{x}^{(0)}$ and the image $\mathbf{x}^{(t)}$ (i.e., generated at iteration t), the algorithm fixes the posterior distribution at iteration t. Refer to Alg. 2 for the proposed algorithm, which is detailed below.

First, we need to build a **transition matrix**, which is estimated from the training set $\mathcal{X}$ by computing the probability distribution $p(y=i|y=j)$ for $i,j \in \{1,...,U\}$. This distribution is estimated assuming that U sets of class specific training images are available. These sets are represented by

Algorithm 2 Closed-loop inference

```
1: Set $\mathbf{x}^{(0)} = \mathbf{x} \in [0,1]^{M \times M}$, representing the input image.
2: Transition matrix: compute transition matrix, representing $p(y = i|y = j)$ (for $i, j \in \{1, ..., U\}$) using (8).
3: for t = 1:T do
4:     Prediction: compute the prediction distribution $p(y^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1})$ with (9).
5:     Data association: search for the image that maximizes $p(\mathbf{x}^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1})$ with (10), by taking $\mathbf{x}^{(t)}$ as defined in (11).
6:     Correction: correct the current prediction by estimating $p(y^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t})$ with (12).
7: end for
8: The class for image $\mathbf{x}$ is the one that maximizes (12).
```

$\mathcal{X}_j = \{\mathbf{x}_n|(\mathbf{x}, y)_n \in \mathcal{X}, y_n = j\}$ (with $j \in \{1, ..., U\}$), and the distribution is estimated as follows:

$$p(y = i|y = j) = (1/Z) \sum_{\mathbf{x} \in \mathcal{X}_j} [(1 - \alpha)p(y = i|\mathbf{x}) + \alpha], \quad (8)$$

where Z is a normalization factor, $p(y = i|\mathbf{x})$ is computed with (6), and $\alpha \in [0,1]$ represents a noise added to this transition matrix that is present in order to avoid an over-confident estimation based solely on the training set. Note that this transition matrix is in fact represented by the confusion matrix computed from the training set results using the classifier in (6), but we use the name "transition matrix" to keep the same nomenclature of visual tracking methods [22].

The **prediction** step estimates the distribution that the inference will change the DBN's current belief about the classification result of the input data $\mathbf{x}$. This is achieved with the following calculation:

$$p(y^{(t)} = i|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1}) = \sum_{j=1}^{U} p(y^{(t)} = i|y^{(t-1)} = j)p(y^{(t-1)} = j|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1}), \quad (9)$$

with the first term of the sum computed with (8) and the second term, with (6) for $t = 1$ and with (12) for $t > 1$.

The **data association** searches for a new image generated by the DBN with the feedback method described in Sec. II-B, that maximizes the following probability:

$$p(\mathbf{x}^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1}) = \sum_{j=1}^{U} p(\mathbf{x}^{(t)}|y^{(t)} = j)p(y^{(t)} = j|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1}), \quad (10)$$

where the second term of the sum is computed with (9), and the first term is computed with (6) using the Bayes theorem and assuming the priors $p(\mathbf{x}^{(t)})$ and $p(y^{(t)} = j)$ are constants. An important part of the data association is how the image $\mathbf{x}^{(t)}$ is generated, especially because as described in Fig. 1, we want the error between the generated image and the original input image to be taken into account in order to fix the current belief state of the DBN. Therefore, the image generation is performed in two steps. First, we sample the distribution $p(y^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1})$ from (9) in order to obtain $\mathbf{u}^{(t)}$ that will be used as the input to generate an image $\tilde{\mathbf{x}}^{(t)}$, as described in the feedback method of Sec. II-B. Second, the actual input image for (10) must have the errors amplified such that the DBN can try to fix the errors, so we have:

$$\mathbf{x}^{(t)} = \max(0, \min(1, \tilde{\mathbf{x}}^{(t)} + \kappa(\mathbf{x}^{(0)} - \tilde{\mathbf{x}}^{(t)}))), \quad (11)$$

where κ is a parameter that influences the amount of error to be included in the image $\mathbf{x}^{(t)}$, $\max(.,.)$ returns the maximum of the two values in the parameters, and $\min(.,.)$ returns the minimum (this guarantees that the final image will be in the range $[0,1]$). Therefore, the idea of this data association stage is to generate a fixed number of images and select the one that produces the highest value for (10).

Finally, the **correction** procedure fixes the current inference by putting together the results from the prediction and data association steps, as follows:

$$p(y^{(t)} = i|\{\mathbf{x}^{(l)}\}_{l=0}^{t}) = (1/Z)p(\mathbf{x}^{(t)}|y^{(t)} = i)p(y^{(t)} = i|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1}), \quad (12)$$

where $Z = \sum_{j=1}^{U} p(\mathbf{x}^{(t)}|y^{(t)} = j)p(y^{(t)} = j|\{\mathbf{x}^{(l)}\}_{l=0}^{t-1})$, the first term is computed in the same way as in (10), and the second term is the prediction from (9).

D. Example

In this section we show examples using the MNIST dataset [1] and the Multiple Object Categories database [2], which are described in more detail in Sec. III-A- III-B. For MNIST, there are several possible sources of errors, such as "7" being confused with "9", or "4" classified as "6", etc. On the other hand, for the Multiple Object Categories dataset, most of the errors are caused by the large shape variation of the Dragonfly class. In Fig. 3 and Fig. 4, we show several examples from both datasets, where the closed-loop inference fixes the result from the feed-forward inference. It is interesting to notice that as the inference evolves, the generated image usually gets more similar to the original image, a fact that is evidenced by the reduced error between the original and generated images.

III. Experiments

The major goal of this section is to show empirical evidence that the closed-loop inference proposed in this paper can improve the results obtained from pure feed-forward approaches based on DBN and also to show how the recognition results are affected by the parameters α in (8) and κ in (11). The number of iterations T in Alg. 2 is fixed at 10 for all experiments below. We show these results using the MNIST database [1] and the Multiple Object Categories Database [2].

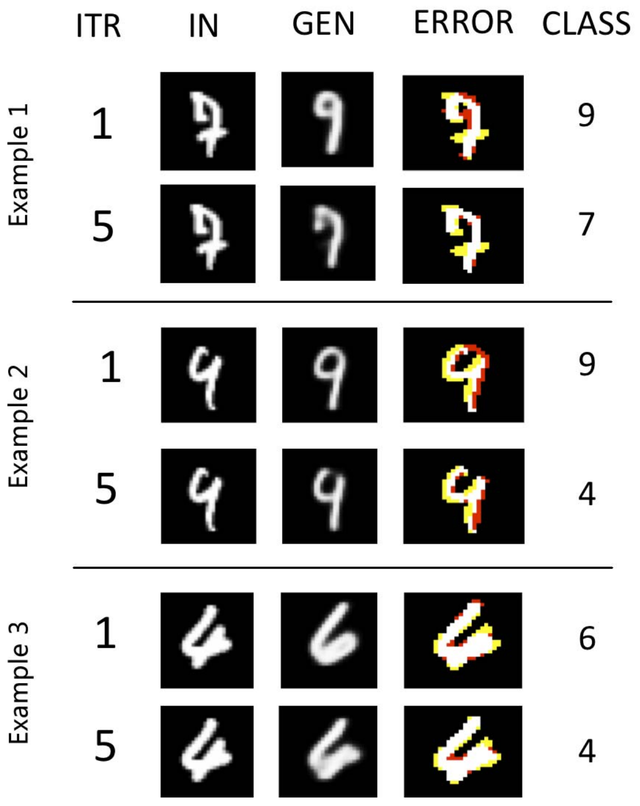

Fig. 3. Examples of the closed-loop inference using the MNIST dataset [1] (see Section III-A). The columns show the iteration index t of the closed-loop inference presented in Alg. 2 (labeled "ITR"), the original input image $\mathbf{x}^{(0)}$ ("IN"), the generated image $\tilde{\mathbf{x}}^{(t)}$ in (11) ("GEN"), the image difference between $\mathbf{x}^{(0)}$ and $\tilde{\mathbf{x}}^{(t)}$ (yellow means bits present in $\mathbf{x}^{(0)}$, but absent in $\tilde{\mathbf{x}}^{(t)}$; and red are bits present in $\tilde{\mathbf{x}}^{(t)}$, but absent in $\mathbf{x}^{(0)}$), and the class producing the maximum value for $p(y^{(t)}|\{\mathbf{x}^{(l)}\}_{l=0}^{t})$ in (12) ("CLASS"), respectively.

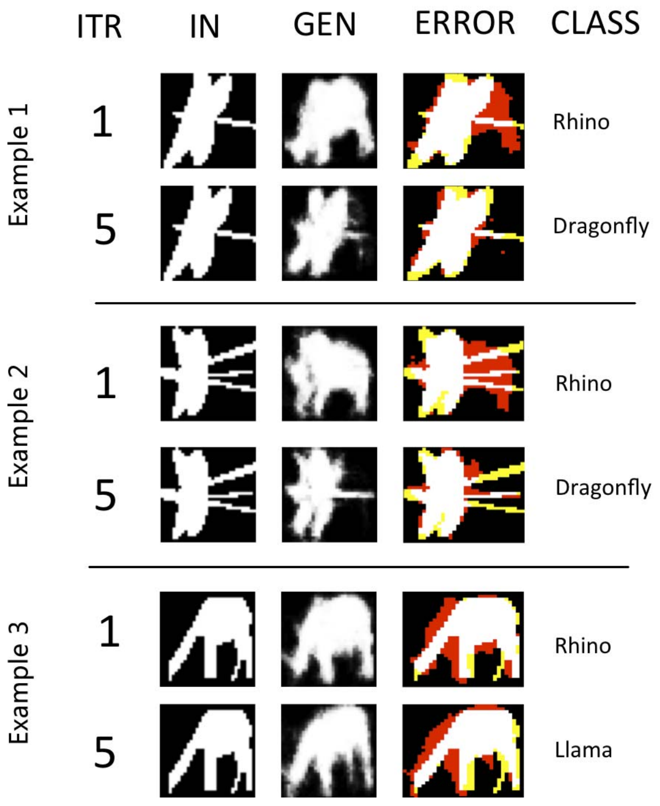

Fig. 4. Examples of the closed-loop inference using the Multiple Object Categories database [2] (see Section III-B). These examples share the same legends of Fig. 3.

A. MNIST

The publicly available MNIST database (see Fig. 5) containing handwritten digits [1] has been used to test several different classification algorithms, which makes it a useful testbed for training and inference algorithms. It contains 60,000 training images and 10,000 test images, and we adopt the framework where no knowledge of the geometry is provided and no special preprocessing or enhancement of the training set is available. The DBN core structure is depicted in Fig. 2 and consists of a visible layer $\mathbf{x}$ with 784 nodes (receiving the 28×28 input images), K hidden layers, with each $\mathbf{h}_k$ containing $\#nodes_k$ nodes, and $\mathbf{u}$ with 10 nodes, representing the labels of each of the 10 classes. The DBN was trained with the 60,000 training images, which were divided into 600 balanced mini-batches, each containing 10 examples of each class. The update of the weights happen after each mini-batch, and we only use the greedy training, where each layer is trained sequentially, as described in Sec. II-A, with 100 epochs. Also, for the top layer training, the labels are provided as part of the input and are represented by turning on one unit in a "softmax" group of 10 units [23] as denoted in (6).

For all results below, we repeat this training process 5 times and report the average error obtained in the test set, which is the percentage of mislabeled test samples. In order to verify the correctness of our implementation, we compute the error achieved by a DBN with the following structure: a visible layer with 784 nodes, 3 hidden layers, with $\mathbf{h}_1$ and $\mathbf{h}_2$ containing 500 nodes, $\mathbf{h}_3$ with 2000 nodes, and $\mathbf{u}$ with 10 nodes. The error for the greedy learning of the DBN above reported by Hinton et al. [23] is 2.49%, while in our case, we obtained 2.51%.

Our first experiment shows the behavior of the proposed closed-loop inference compared with the feed-forward inference as a function of the parameters α in (8) and κ in (11) and also in terms of different DBN structures. Figure 7 shows how the error is affected in percentage terms with different values for α and κ using the DBN structure described by Hinton et al. [23]. Specifically, the number of elements in brackets represent the number of hidden layers, and each number in sequence denotes the number of nodes in each layer. First notice that for the shown structure, the closed-loop inference can provide a relative improvement of between 2% and 8% on average for the five runs considered. In fact, for some of those runs, we can achieve an improvement of over 20%. Second, it is clear that the parameter κ has a stronger influence in the

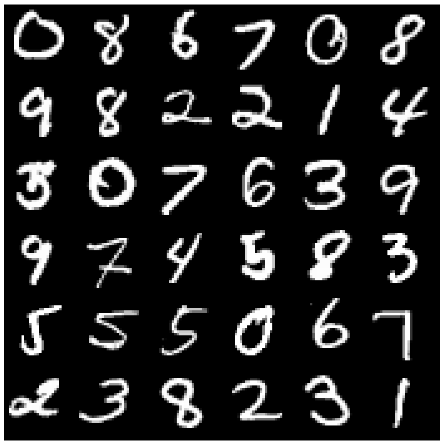

Fig. 5. Samples from the MNIST [1] database.

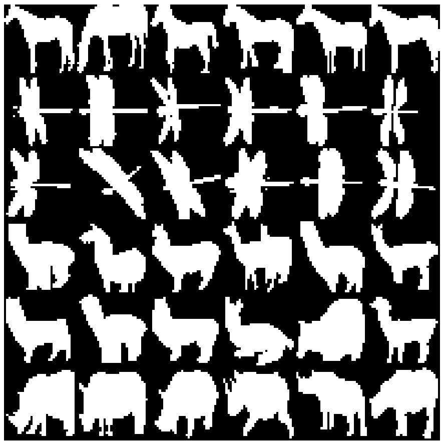

Fig. 6. Samples from the Multiple Object Categories [2] database.

result than α, particularly when the DBN has a more complex structure (e.g., see the bottom graphs). Finally, as the DBN structure becomes more complex, the behavior of the closed-loop inference (w.r.t. to the parameters studied) becomes more predictable, which means that it alleviates the need to perform model validation methods (e.g., cross-validation).

For the second experiment, we fix $\alpha = 0.5$ and $\kappa = 2.0$ and plot the average error (over the 5 test runs explained above) as a function of the complexity of the DBN structure both for the feed-forward and closed-loop inferences (see Fig. 8). Notice that the closed-loop inference always improves the result produced by the feed-forward inference, and the best error achieved using the same structure as in [23] is 2.30%. In order to achieve more competitive results for that specific database [24], we re-train the DBN with 1000 epochs (instead of 100) using Alg. 1, and use it to initilize the up-down training procedure (we ran 300 epochs of this learning algorithm), which is a slower estimation procedure that fine-tunes the weights [23]. With this new training, we achieved an error of 1.22% using feed-forward inference, and 1.16% with our proposed closed-loop inference.

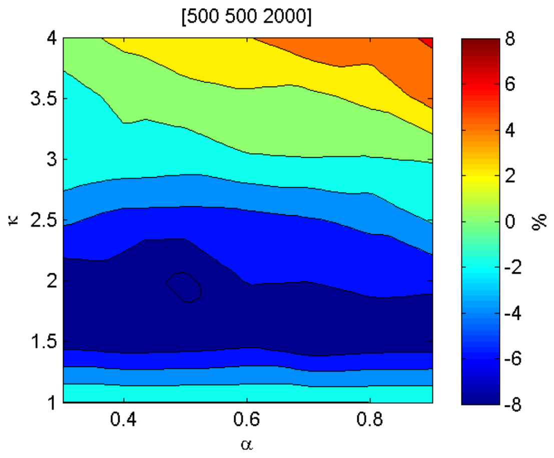

Fig. 7. Error reduction and increase with the use of closed-loop inference as a function of the parameters α in (8) and κ in (11). The graph shows the result obtained from the DBN structure with $\mathbf{h}_1$ and $\mathbf{h}_2$ containing 500 nodes, $\mathbf{h}_3$ with 2000 nodes.

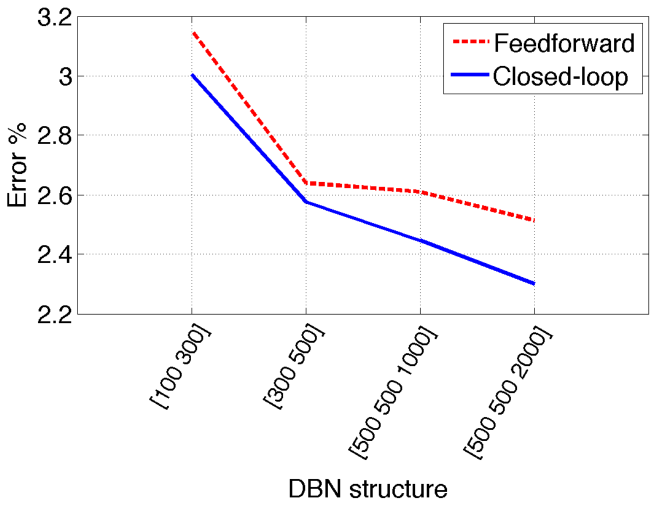

Fig. 8. Average error results as a function of the DBN structure for the feed-forward and closed-loop inferences, considering that $\alpha = 0.5$ and $\kappa = 2.0$.

B. Multiple Object Categories Database

Recently, Eslami et al. [2] presented the shape Boltzmann machine (shapeBM), where they tested their method on a database containing four visual classes, comprising (see Fig. 6): 1) 328 horse images facing left, but in a variety of poses (i.e., the Weizmann horses [25]); 2) 68 dragonfly images [26]; 3) 78 llama images [26]; and 4) 59 rhino images [26]. These 533 images are cropped and normalized to have size 32×32 pixels. Also, we use same number of layers and nodes per layer as the methodology described by Eslami et al. [2], which is as follows: two layers of latent variables, with $\mathbf{h}_1$ containing 2000 nodes and $\mathbf{h}_2$ with 400 units. The main differences of our approaches (in terms of the DBN structure) are: the introduction of the layer $\mathbf{u}$ with

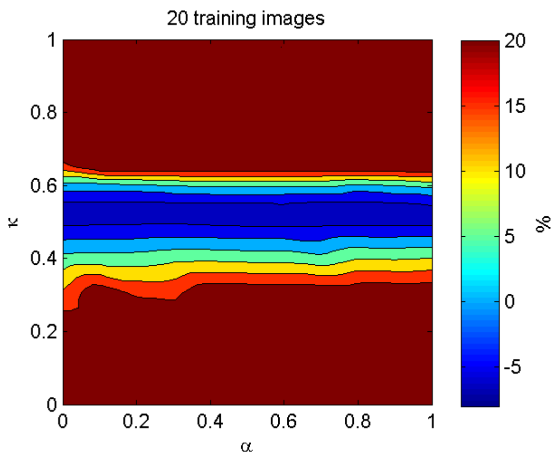

Fig. 9. Example of performance on the Multiple Object Categories Database using 20 training images. The performance of the closed-loop inference compared to the feed-forward inference as a function of the parameters α in (8) and κ in (11).

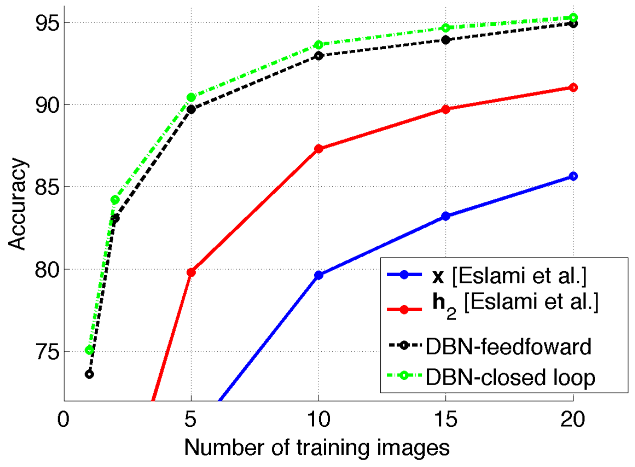

Fig. 10. Comparison of the performances of the classification methods described in [2] (red and blue curves), the feed-forward inference (dotted black) and the closed-loop inference (dotted green).

4 nodes (representing the label of each of the 4 classes), and the full connection between the nodes of the visible and first hidden layer. The training is performed in the same way as for the case of the MNIST database, with the exception that we had 1000 epochs and that we augment the training set by applying random rigid deformations to the original images. This is particularly important for the experiments that use an extremely small number of training images. In fact, the number of training images per class for this experiments varied from $R = 1$ to 20, with $59 - R$ test images per class (note that the training and test images are picked randomly from the sets described above). We average the accuracy and error results over 100 runs, where each run involves a training, and feed-forward and closed-loop inferences.

We first show the performance of the closed-loop inference compared to the feed-forward inference as a function of the parameters α in (8) and κ in (11) and in terms of the number of training images. In Fig. 9, we show the percentage reduction and increase of the error using the closed-loop inference. Notice that for this database, it is clear that α is not as important as the parameter κ. Also, notice that on average the closed-loop inference reduces the error by 5%. Figure 10 compares the accuracy results obtained by the classification method described by Eslami et al. [2] (see red and blue curves) with our feed-forward (dotted black) and closed-loop inferences (dotted green). Notice that the feed-forward inference produces competitive results, and actually improves the results by Eslami et al. [2]. Also, the closed-loop inference improves upon the results of the feed-forward inference. It is worth mentioning that this experiment provides evidence of one of our claims, which is that the closed-loop inference can be used in order to fix the mistakes of classifiers trained with few training images. This is particularly clear for the cases with 1 to 10 training images, where the relative improvement is in the range of 5% of the accuracy result obtained with feed-forward analysis.

IV. Discussion and Conclusions

The idea of using some intermediate form of the analysis to drive some sort of feedback mechanism has been a recurrent topic in the field of computer vision. There has been a renewed interest in this topic lately, and we believe that the development of DBNs with their discriminative and generative capabilities will be quite important for the further exploration of this topic. In this paper we propose the first closed-loop inference mechanism based on DBNs. Using databases of handwritten digits and shapes, we show that our methodology always improves the accuracy of feed-forward mechanisms in a range that varies from 5% to 10%, and we show the best results in the field for the Multiple Object Categories database [2]. For the MNIST database, we present competitive results compared to DBN results using a similar network structure and training setup, but we realize that our results are not comparable to the best proposed approaches in field [24], which show error results in the order of 0.2%. We will investigate further how we can obtain results in this range with the proposed closed-loop inference. Furthermore, it is interesting to study the role of α, which in essence tells how much we trust the transition matrix in (8). From the experiments, α seems to matter only when the DBN structure offers limited capacity to model the problem properly, which is a fact that can be used in order to do model selection. Finally, with the Multiple Object Categories database, we show that our closed-loop inference is able to fix mistakes made by the classifiers trained with small training sets, which is one of the issues present in current approaches, as explained in the introduction.

It can be noticed from the experiments that the accuracy of the presented closed-loop inference is highly associated with the accuracy of the DBN feed-forward inference. We will investigate how our proposed closed-loop inference can be adapted to other types of neural networks, such as Deep Boltzmann Machines [27], and other types of open loop classifiers, such as Support Vector Machine [28].

One of the major points not covered by this paper is the ability of our approach to generate gray-level images in addition to the black and white images shown above. The main issue is that the generation of more realistic gray-level images requires much more complex DBN structures [20], and it is not yet clear whether these DBNs will be able to generate images that are detailed enough to be used by our feedback mechanism. Another problem is with respect to the scalability of our mechanism in terms of the number of classes. For instance, as the number of classes increases, it has been shown that it is possible to build powerful classifiers with deep belief networks. For instance, the recently propose DBN used for the classification of 1000 Imagenet classes [29] presents competitive recognition results. However, it is not clear if such complex network will be able to generate representative images of specific visual classes. Finally, another problem with the use of our approach is with respect to its efficiency because, even though the feed-forward inference is reasonably fast, the simple use of our approach implies an inference that is as slower as the number of iterations required by the closed-loop mechanism. However, recent implementations of DBN make full use of GPU programming [29], which means that this issue can be considerably alleviated.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1, 2, 4, 5, 6

[2] S. A. Eslami, N. Heess, and J. Winn, "The shape boltzmann machine: a strong model of object shape," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 406–413. 1, 2, 4, 5, 6, 7

[3] H. Barrow and J. Tenenbaum, "Computer vision systems," in *Computer vision systems: papers from the Workshop on Computer Vision Systems, held at the University of Massachusetts, Amherst, Massachusetts, June 1-3, 1977*. Academic Pr, 1978, p. 3. 1, 2

[4] I. Biederman, "Human image understanding: Recent research and a theory," *Computer Vision, Graphics, and Image Processing*, vol. 32, p. 2973, 1985. 1, 2

[5] T. Binford, "Visual perception by computer," in *IEEE Conference on Systems and Control*, 1971. 1, 2

[6] R. Brooks, "Model-based 3-d interpretations of 2-d images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 140–150, 1983. 1, 2

[7] S. Dickinson, *Evolution of Object Categorization and the Challenge of Image Abstraction*. Cambridge University Press, 2009, in: S. Dickinson, A. Leonardis, B. Schiele, and M. Tarr, (eds), Object Categorization: Computer and Human Vision Perspectives. 1

[8] U. Grenander and M. Miller, *Pattern Theory: From Representation to Inference*. Oxford University Press, 2007. 1

[9] D. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artificial Intelligence*, vol. 31, no. 3, pp. 355–395, 1987. 1, 2

[10] A. Pentland, "Perceptual organization and the representation of natural form," *Artificial Intelligence*, vol. 28, pp. 293–331, 1986. 1, 2

[11] S. Zhu and D. Mumford, "A stochastic grammar of images," *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 4, pp. 259–362, 2006. 1, 2

[12] M. J. Choi, A. Torralba, and A. Willsky, "A tree-based context model for object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 240–252, 2012. 1

[13] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, 2010. 1

[14] Y. Lee and K. Grauman, "Object-graphs for context-aware visual category discovery," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 346–358, 2012. 1

[15] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding:classification, annotation and segmentation in an automatic framework," in *Proc. IEEE Comp. Vision and Pattern Recognition*, 2009. 1

[16] D. Marr, "Representing visual information," *Computer vision systems*, pp. 61–80, 1978. 1, 2

[17] A. Barriuso and A. Torralba, "Notes on image annotation," *arXiv preprint arXiv:1210.3448*, 2012. 1

[18] D. Hoiem, A. A. Efros, and M. Hebert, "Closing the loop in scene interpretation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8. 1, 2

[19] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, p. 1, 2010. 1, 3

[20] M. Ranzato, J. Susskind, V. Mnih, and G. Hinton, "On deep generative models with applications to recognition," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2857–2864. 2, 8

[21] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. 3

[22] Y. Bar-Shalom, *Tracking and data association*. Academic Press Professional, Inc., 1987. 3, 4

[23] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. 5, 6

[24] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649. 6, 7

[25] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*. IEEE, 2004, pp. 46–46. 6

[26] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*. IEEE, 2004, pp. 178–178. 6

[27] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455. 8

[28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995. 8

[29] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114. 8