COSMO: Contextualized Scene Modeling with Boltzmann Machines

İlker Bozcan*, Sinan Kalkan*

*KOVAN Research Lab, Dept. of Computer Engineering, Middle East Technical University, Ankara, Turkey

Abstract

Scene modeling is very crucial for robots that need to perceive, reason about and manipulate the objects in their environments. In this paper, we adapt and extend Boltzmann Machines (BMs) for contextualized scene modeling. Although there are many models on the subject, ours is the first to bring together objects, relations, and affordances in a highly-capable generative model. For this end, we introduce a hybrid version of BMs where relations and affordances are introduced with shared, tri-way connections into the model. Moreover, we contribute a dataset for relation estimation and modeling studies. We evaluate our method in comparison with several baselines on object estimation, out-of-context object detection, relation estimation, and affordance estimation tasks. Moreover, to illustrate the generative capability of the model, we show several example scenes that the model is able to generate.

Keywords: Scene Modeling, Context, Boltzmann Machines.

1. Introduction

Having a model, i.e., a representation, of the environment is essential for artificial and biological cognitive agents. A scene model is a representation that allows a robot to reason about the scene and what it contains in an efficient
manner. For example, as shown in Figure 1(a), using a scene model, a robot can check (i) whether there is a certain object in the scene and if yes, where it is; (ii) whether an object is in the right-place in the scene; or (iii) whether there is something not expected or redundant in the scene.

A contextualized scene model, on the other hand, integrates the context of the scene in representing the environment and making inferences about what it contains. This is critical since it has been noted that context plays critical role in perception, reasoning, communication and action [1][2]. Context helps these processes in resolving ambiguities, rectifying mispredictions, filtering irrelevant details, and adapting planning. These processes and problems are closely linked
to a scene model, and therefore, scene models should contextualize what they represent.

In this paper, we use higher-order Boltzmann Machines [3, 4] for the first time for contextualized scene modeling. In our model, as shown in Figure 1, objects, spatial relations between them and affordances are considered as visible units. The hidden (latent) units then represent high-order co-occurrence relations among the visible units, i.e., contextual information about the scene and what it contains. See Section 2.4 for a more detailed analysis of our contributions.

Although there are many studies on scene modeling in robotics, ours is the first to use (and adapt) Boltzmann Machines for scene modeling, which not only represents objects or relations between objects in the scene but also affordances of objects. Moreover, being generative, it is able to complete any missing information in the scene and make predictions given any information that may be available.

2. Related Work

In this section, we review related work on scene modeling, relation estimation and affordance estimation.

2.1. Scene Modeling

Scene modeling is an important problem in Computer Vision and Robotics. During the last decade, especially probabilistic methods or probabilistic graphical models such as Markov Random Fields or Conditional Random Fields [6, 8, 16, 17], Bayesian Networks (BN) [13, 20], Latent Dirichlet Allocation variants (LDA v.) [14, 15], Dirichlet and Beta (DB) processes [5], chain-graphs [11], predicate logic (PL) [9, 12], Scene Graphs [17], and ontologies [12, 13, 19] have been proposed for solving the problem.

Among these studies, similar to ours, there are also models that explicitly integrate context into the model [14, 15, 17]. For example, Wang et al. [14]
extend LDA to incorporate spatial positions between pixels in a local neighborhood in order to segment an image into semantically meaningful regions. Philbin et al. [15], on the other hand, include spatial arrangement between visual patches (i.e., words in LDA) to group similar images into a topic.

Among these, the work of Çelikkanat et al. [17] is the closest to ours. Çelikkanat et al. use object detections as visible variables and context as the latent variable in LDA. However, in their work, the main focus was on incremental learning of context nodes, and issues like spatial relations and generative abilities of the scene model were not considered.

### 2.2. Relation Estimation and Reasoning

Without loss of generality, we can broadly analyze relation estimation and reasoning studies in three main categories: The first category of methods use hand-crafted rules to determine whether a pre-determined set of spatial relations...
were present between objects in 2D or 3D, e.g., [21].

In the second category of methods, which use probabilistic graphical models such as Markov Random Fields [6, 10], Conditional Random Fields [16], Implicit Shape Models [22], and latent generative models [5], a probability distribution is modeled for relations between objects or entities. In these studies, Anand et al. [6] considered relations like “on-top” and “in-front” (and their symmetries); Celikkanat et al. [10], “left”, “on”, and “in-front” (and their symmetries); Lin et al. [16], “on-top”, “close-to” relations; Meissner et al. [22], 6-DoF relations (rotation and translation) between objects; and, Joho et al. [5], an implicit model over local arrangements of objects is learned.

In the third category, relation estimation is formulated as a classification problem and solved using discriminative models, such as logistic regression [23], and deep learning [24]. The study by Guadarrama et al. [23] studied relations like “above”, “behind”, “close to”, “inside of”, “on”, and “left” (and their symmetries), whereas only two relations (“left”, “behind” - and their symmetries) are considered in [24].

Existing efforts on modeling or estimating relations generally address the problem either for relations or relations and objects, and not consider related concepts such as affordances. Moreover, Boltzmann Machines have not been used for the problem in a scene modeling context.

2.3. Affordance Prediction

The concept of affordance, owing to Gibson [25], pertains to the actions that are provided by entities in the environment to the agents. With suitable formalisms for robotics studies [26], affordance-based models have been proposed for many important problems, such as manipulation [27], navigation [28], imitation learning [29], planning [30, 31], conceptualization [31, 32], – see [33, 34] for a review.

An important issue in affordance-based approaches is to be able to estimate the affordances of objects from visual input. For this end, support vector machines [30, 35], bayesian networks [36, 37], markov random fields [8, 38], and
deep networks [39] [40] [41] have been widely used in the literature. However, affordance prediction is generally addressed independently from scene modeling tasks, and to the best of our knowledge, Boltzmann Machines have not been used for modeling affordances.

2.4. Contributions of the Current Study

Looking also at the summary of the existing studies in Table 1, we see the following as the main contributions of the current paper:

- To the best of our knowledge, ours is the first to use Deep Boltzmann Machines (DBM) [42] for scene modeling. With DBM, we introduce a generative scene model which incorporates objects, spatial relations and affordances.

- In order to be able to model concepts like relations and affordances that require tri-way connections, we adapt and extend DBM by (i) combining together General BM [3] with higher-order BM [4], and (ii) introducing weight-sharing in order to have the same concepts of relations and affordances between different sets of variables.

The code and the dataset are publicly made available at: https://github.com/bozcani/COSMO

We apply our model on relevant robot problems: Determining (i) what is missing in a scene, (ii) relations between objects, (iii) what should not be in a scene, (iv) the affordances of objects, and (v) generating novel scenes given some objects or relations from the to-be-generated scene. We compare our model (COSMO) against DBM [42] with 2-way relations (GBM), and Restricted Boltzmann Machines (RBM) [43].

The current paper extends our previous work accepted for a conference [44]. To be specific, the current paper extends the model by also including affordances, performs a more rigorous investigation of the proposed model with extensive experiments on the detailed investigation of the architecture and ex-
periments with another dataset (namely, visual genome [45]). Moreover, the current paper includes experiments with a real robot.

3. Background: Boltzmann Machines

![Boltzmann Machines Diagram]

Figure 2: An illustration of different types of Boltzmann Machines (BM): General BM, Restricted BM and Deep BM. BM is stochastic network that is able to model probability distributions of high-dimensional data, and therefore, generate novel samples.

A Boltzmann Machine (BM) [3] is a stochastic, generative network. A BM can model the probability distribution of data, denoted by \( v \), with the help of hidden variables, \( h \):

\[
p(v) = \sum_h p(v, h).
\]

(1)

In BMs, \( v = \{v_i\}_{i=1}^V \subset \{0, 1\}^V \) is called the set of visible nodes, and \( h = \{h_i\}_{i=1}^H \subset \{0, 1\}^H \) the hidden nodes. The visible nodes and the hidden nodes are connected to each other and how they are connected have led to different models – see Figure 2. In BMs, the connections are bi-directional; i.e., information can flow in both directions.

\[\text{This section is necessary for explaining our method, although what it covers is textbook material.}\]
In a BM, one can talk about the compatibility, i.e., harmony, between two nodes connected by an edge. If, e.g., \( n_iw_{ij}n_j \) is high for two nodes connected by an edge with weight \( w_{ij} \), then nodes \( n_i \) and \( n_j \) are more compatible. However, generally, in BMs, the negative harmony, i.e., the energy of the network is used:

\[
E(v, h) = -\sum_{i<j} v_iw_{ij}v_j - \sum_{i<j} h_iw_{ij}h_j - \sum_{i<j} h_iw_{ij}h_j,
\]

(2)

where \( w_{vv}, w_{hh} \) and \( w_{hv} \) are the weights of the edges connecting visible-visible nodes, hidden-hidden nodes, and hidden-visible nodes respectively.

Being inspired from statistical mechanics, where systems with lower energies are favored more, BM associates the probability of being in a state (i.e., a configuration of \((v, h)\)) with the energy of the system as follows:

\[
p(v, h) = \frac{1}{Z} \exp(-E(v, h)),
\]

(3)

where the normalizing term, also called the partition function, is defined as: \( Z = \sum_{v', h'} E(v', h') \). Notice that \( Z \) requires an integration over all possible states of the system, which is impractical to calculate in practice. Therefore, \( p(v, h) \) is iteratively learned by stochastically activating nodes in the network with probability based on the change in the energy of the system for an update:

\[
p(n = 1) = \frac{1}{1 + e^{\Delta E_n/T}},
\]

(4)

where \( n \) is a visible or a hidden node; \( \Delta E_n \) is the change in energy of the system if node \( n \) is turned on; and \( T \) is the temperature of the system, gradually decreased (annealed) to a low value. When \( T \) is high, the system can make radical updates that can even increase its energy; and when \( T \) is lowered, Equation 4 forces the network to make more deterministic updates, which lower the energy of the system.

3.1. Training a BM

Training a BM means that its weights are updated to model \( p(v) \) as accurately as possible. Let us use \( p^+(v) \) to denote the true probability of the data,
and $p^-(v)$, the probability estimated by the model. Then, a BM is trained in order to minimize the dissimilarity, e.g., the Kullback-Leibler divergence, between $p^+(v)$ and $p^-(v)$. Taking gradient of the divergence with respect to a weight, $w_{ij}$, gives us the rate at which we should update it:

$$w_{ij} \leftarrow w_{ij} - \alpha (p^+_{ij} - p^-_{ij}),$$

where $p^+_{ij}$ is the expected joint activation of nodes $s_i$ and $s_j$ when samples from the data are clamped on the visible units and the state of the network is updated accordingly (called the positive phase); $p^-_{ij}$ is the expected joint activation of nodes $s_i$ and $s_j$ when the network is randomly initialized and the state of the network is updated accordingly (called the negative phase); and $\alpha$ is a learning rate.

For training BMs, maximum Likelihood based methods are used \cite{3, 42, 46}. However, since the partition function, $Z$, is intractable, directly computing $p^+_{ij}$ and $p^-_{ij}$ is not possible for general BMs. Therefore, Monte Carlo Markov Chain methods such as Gibbs sampling or Variational Inference methods such as mean field approaches are used to approximate $p^+_{ij}$ and $p^-_{ij}$. Despite these methods, learning is still impractical owing to the connections within hidden and visible nodes, and potentially high number of hidden nodes.

### 3.2. BM Variants

Since training is rather slow and limiting in BM, its restricted version (Restricted Boltzmann Machines) with only connections between hidden and visible nodes have been proposed \cite{43}. In a Deep Boltzmann Machine \cite{42}, on the other hand, there are layers of hidden nodes. See Figure 2 for a schematic comparison of the alternative models.

Some problems require the edges to combine more than two nodes at once, which have led to the Higher-order Boltzmann Machines (HBM) \cite{4}. With HBM, one can introduce edges of any order to link multiple nodes together.
4. COSMO: A Contextualized Scene Model with Triway BM

We extend and adapt DBM for contextualized scene modeling task. As shown in Figure 4, our model consists of visible (input) layer and hidden layer(s) corresponding to contextual representation of the scene.

We define a scene \( s \in S \) to be the tuple of an object vector \( o \) describing objects currently visible to the robot, the vector of the spatial relations \( r \) between the objects, and the vector of affordances \( a \). A visible node corresponds to an object, a relation or an affordance, and is set active (value 1) if corresponding object, affordance or relation exists in the scene (in this sense, \( v = (o, r, a) \)). The hidden nodes \( h \) then represent latent joint configurations of the visible nodes; i.e., they correspond to contextual information eminent in the scene.

Relation and affordance nodes link two object nodes with single tri-way edges, and visible nodes are fully connected to hidden nodes \( h \). The overall energy of the hybrid BM then is updated as follows:

\[
E(o, h, r, a) = - \sum_{i<j} h_i w_{ij}^h o_j - \sum_{i,j,k} w_{ijk}^r r_{ijk} o_j - \sum_{i,j,k,l} w_{ijk}^h h_l - \sum_{i,j,k} w_{ijk}^a a_{ijk} o_j - \sum_{i,j,k,l} w_{ijk}^h a_{ijk} h_l,
\]

where the new terms compared to the energy definition in Equation 2 are highlighted in red. \( r_{ijk} \) denotes spatial relation node with type \( i \), between object nodes \( o_j \) and \( o_k \). \( a_{ijk} \) is an affordance relation with type \( i \), between objects nodes \( o_j \) and \( o_k \). \( w_{ijk}^r \) is the weight of the tri-way edge connecting object nodes \( o_j, o_k \) and spatial relation node (visible) \( r_i \); and, similarly, \( w_{ijk}^a \) is the weight of the tri-way edge connecting object nodes \( o_j, o_k \) and affordance node (visible) \( a_i \).

4.1. Training and Inference

In order to make training faster, we dropped the connections between the hidden neurons and took gradient of the divergence (\( KL(p^+(o, r, a) \parallel p^-(o, r, a)) \))
with respect to each type of weight as in Equation 5.

According to the new energy definition (Equation 7) and connections, probabilities of being active for visible and hidden units are given by:

\[
p(o_i = 1 \mid o, h, r, a) = \sigma \left( \sum_j h_j w_{ij}^{ho} + \sum_{j,k} w_{ijk}^{r} r_{ijk} o_j + \sum_{j,k} w_{ijk}^{a} a_{ijk} o_j \right),
\]

(7)

\[
p(h_l = 1 \mid o, r, a) = \sigma \left( \sum_i o_i w_{il}^{ho} + \sum_{i,j,k} r_{ijk} w_{rij}^{r} + \sum_{i,j,k} a_{ijk} w_{ijkl}^{a} \right),
\]

(8)

\[
p(r_{ijk} = 1 \mid o, h) = \sigma \left( w_{ijk}^{r} o_j o_k + \sum_{l} w_{ij}^{r} h_l \right),
\]

(9)

\[
p(a_{ijk} = 1 \mid o, h) = \sigma \left( w_{ijk}^{a} o_j o_k + \sum_{l} w_{ij}^{a} h_l \right).
\]

(10)

For training COSMO, in the positive phase, as usual, we clamp the visible units with the objects, the relations and the affordances between the objects and calculate \(p^+\) for any edge in the network.

In the negative phase, firstly, object units are sampled with a two-step Gibbs sampling by using activation of hidden units only. In this way, initially, the model sees the environment as bag of objects by not considering relations and affordances. Then, relation and affordance nodes are sampled by using hidden nodes (context) and recently sampled object nodes. We calculate \(p^-\) for any edge in the network with these two steps.

The overall algorithm is summarized in Algorithm 1.

At the end of the negative phase, input scene \((s)\) is re-sampled, and \(s'\) denotes new scene including recently sampled objects, relations and affordances during negative phase.

Since our dataset has small number of samples and input vectors are too sparse, precise inferences are crucial. Therefore, we prefer Gibbs sampling [47] that is a Monte Carlo Markov Chain (MCMC) method to approximate true data distribution instead of variational inference since MCMC methods can provide precise inference but variational inference methods cannot guarantee that [48].
Algorithm 1 Training COSMO.

1: **Input:** Training data, \( S = \{s^i\}_i \); learning rate, \( \alpha \); number of epochs, \( m \).

2: **Output:** Learned weights, \( w \).

3: 

4: for \( m \) epochs do

5: for \( s \in S \) do

6: /* Positive Phase */

7: \( o^{(0)} \leftarrow s_o^a, r^{(0)} \leftarrow s_r^a, a^{(0)} \leftarrow s_a^a \),

8: \( h^{(0)} \leftarrow p(h \mid o^{(0)}, r^{(0)}, a^{(0)}) \)

9: Calculate \( p^+ \) for each edge.

10: /* Negative Phase */

11: Sample \( \hat{h}^{(0)} \) using Eqn. 8.

12: \( o^{(1)} \leftarrow 0, r^{(1)} \leftarrow 0, a^{(1)} \leftarrow 0 \)

13: \( o^{(1)} \leftarrow p(o \mid o^{(1)}, \hat{h}^{(0)}, r^{(1)}, a^{(1)}) \)

14: \( r^{(1)} \leftarrow p(r \mid o^{(1)}, \hat{h}^{(0)}, a^{(1)}) \)

15: \( a^{(1)} \leftarrow p(a \mid o^{(1)}, \hat{h}^{(0)}, r^{(1)}) \)

16: Sample \( \hat{o}^{(1)}, \hat{r}^{(1)}, \hat{a}^{(1)} \) using Eqn. 7, 9, 10.

17: \( h^{(1)} \leftarrow p(h \mid \hat{o}^{(1)}, \hat{r}^{(1)}, \hat{a}^{(1)}) \)

18: Calculate \( p^- \) for each edge.

19: 

20: Update weights using Eqn. 5.

5. Experiments and Results

In this section, we evaluate COSMO on several scene modeling and robotics problems and compare the model against several baselines and alternative methods whenever possible.

5.1. The Dataset

For our experiments, we formed a dataset composed of 6,976 scenes, half of which is sampled from the Visual Genome (VG) dataset and the other half
from the SUN-RGBD dataset. We used samples from both datasets since (i) the VG dataset has spatial relationships but these do not include relations useful for robots, such as left and right, and (ii) the VG dataset mostly includes outdoor datasets, which we compensate using the SUN-RGBD dataset, which is composed of indoor scenes only. Therefore, we included equal number of samples from both the VG and the SUN-RGBD datasets. However, the SUN-RGBD dataset did not have spatial relations labeled, therefore, we did manual labeling for the SUN-RGBD dataset.

Our dataset consists of 90 objects that commonly exist in scenes, including human-like (man, woman, boy etc.), physical objects (cup, bottle, jacket etc.), part of buildings (door, window etc.).

Our dataset is composed of the following eight spatial relations: left, right, front, behind, on-top, under, above, below. These spatial relations are annotated in the VG dataset already. However, we extended the original SUN-RGBD dataset by manually annotating these eight spatial relations. Moreover, we included verb-relations in the VG dataset as affordances into the dataset. The set of affordances include eat-ability, push-ability, play-ability, wear-ability, sit-ability, hold-ability, carry-ability, ride-ability, push-ability, use-ability.

Let us use $S = \{s_1, ..., s_{976}\}$, where $s_i$ denotes $i^{th}$ sample, to denote the dataset. $s_i$ has a vector form that represents the presence of objects, relations and affordances among them in the scene. Active (observed) variables are set to value 1, or to value 0 otherwise. Opposite spatial relations (e.g., left and right) can be represented as single relations in BMs since if object $o_1$ is to the left of object $o_2$, then we can state that object $o_2$ is to the right of object $o_1$. As a result, each sample is represented by a binary vector that has length 113,490 ($90 + 14 \times 90 \times 90$).

5.2. Compared Models

We compare COSMO with General Boltzmann Machine (GBM), Restricted Boltzmann Machine (RBM) and Relational Network (RN) for several scene reasoning tasks that are crucial for various robotic scenarios. For GBM and
RBM models, we used the same number of hidden nodes as in COSMO – see Section 5.4.

5.2.1. General Boltzmann Machine (GBM)

GBMs are unrestricted in terms of connectivity or the hierarchy in the network (either among the hidden or the visible nodes). However, this may make learning impractical, especially when hidden nodes are connected to each other. We allow connections within visible nodes to incorporate interactions between objects as required for scene modeling. In this structure, visible nodes consist of object, relation and affordance nodes as in COSMO. Unlike COSMO, GBM uses two-way edges for relation and affordance nodes, instead of tri-way edges. Similar to COSMO, all visible nodes are fully connected to the hidden nodes but connections within hidden nodes are not allowed. Therefore, the only difference between COSMO and the GBM model is how relation and affordance nodes are connected to objects. To make it comparable with COSMO, we used the same number of layers and hidden neurons in GBM as in COSMO.

5.2.2. Restricted Boltzmann Machine (RBM)

Different from GBM, an RBM only allows connections between visible and hidden nodes. To make it comparable with COSMO, we used the same number of layers and hidden neurons in RBM as in COSMO.
5.2.3. Relation Network (RN)

RNs [50] are simple neural networks to address problems related to relational reasoning. We modified RNs as shown in Figure 4 to make them compatible for our experiments: The input vectors are embedded with a Multi-Layer-Perceptron (MLP) and the activations of MLP are used as object pairs for another MLP, called the $g$ network. In the original model, object pairs are concatenated with an embedding of a query text; however, we omit this since we assume that the model has one type of question for each scenario. For example, for the spatial relation estimation task, only object and affordance vectors are used as input and spatial relations are predicted. In this case, the model is trying to answer the question “what are relations among all objects in the scene?”.

Training RN to answer a specific question “what is the relation between object $a$ and object $b$” requires additional training samples, including question-answer pairs. These are crucial drawbacks of RN when compared to COSMO. Being generative, COSMO have more flexibility on what can be queried with the scene modeled.

Our implementation of the RN method closely follows the original study. However, we had to adjust the architecture to fit to our data sizes. The embedding MLP network is composed of 2 layers (with 128 and 128 neurons respectively) with ReLU non-linearities. The $g$ network is a MLP with 2 layers (with 256 and 256 neurons respectively) with sigmoid non-linearities. The prediction network, $f$, then is a MLP with 2 layers (with 64 and 64 neurons respectively) with sigmoid non-linearities.

We trained the RN model using Adam optimizer with default parameters (and learning rate of 0.001), 32 sized-batches and early stopping.

5.3. Network Training Performance

The dataset (composed of 6,976 scenes) is split into three randomly: 60% for training, 30% for testing and 10% for validation. This split is used for training and testing all methods. For evaluating the training performance, we calculated an error on the difference between the clamped visible units and reconstructed
visible states that are sampled in the negative phase:

\[ E_{\text{train}} = \frac{1}{|S|} \sum_{s \in S} \sum_{i} (p(s_i^+) - p(s_i^-))^2, \tag{11} \]

where the cumulative sum is normalized with the total number of samples (|S|).

Figure 5 plots the error separately for the objects (o), the spatial relations (r) and affordances (a). From the figure, we observe that the error is consistently decreasing for all types of visible units for both the training data and the validation data, suggesting that the networks is learning to represent the probability over objects, the spatial relations and the affordances very well.

However, we observe in Figure 5(b) that the network learns affordances faster than objects and relations. This difference is owing to the fact that the set of possible affordances in a scene is much sparser than objects and relations, making the network quickly learn to estimate 0 (zero) for affordances, leading to a sudden decrease in the loss.

5.4. Analyzing the Hyper-parameters

We evaluated the effect of various hyper-parameters on COSMO’s training performance (Eqn. 11). For all the analyses performed in this section, we looked...
at the error on the validation data (see Section 5.1).

First, we analyzed the effect of the number of hidden layers. For this end, we tested models with 1, 2, 3, 4 and 5 hidden layers. As shown in Figure 5, the model has the lowest reconstruction error (for all types of visible nodes) with only one hidden layer, and the error rises when number of hidden layers increases. This increase might be because the dataset might be falling insufficient for the increase in the number of parameters when the number of layers increases.

Secondly, we analyzed the effect of the number of hidden neurons in a hidden layer. We tested 50, 100, 200, 400, 800 and 2000 hidden neurons. As shown in Figure 6, the reconstruction error decreases when number of hidden neurons increases, as expected.
Table 2: Average running time in seconds (time) for one epoch and total number of parameters for different models.

|                  | COSMO | GBM  | RBM  | RN   |
|------------------|-------|------|------|------|
| Time (seconds)   | 181.06| 208.22| 179.06| 102.30|
| # of params.     | 13,802,600 | 13,871,200 | 13,734,000 | 12,463,104 |

Lastly, we analyzed the effect of different annealing schedules. We tried the following annealing schedules (selected from [51]), namely, exponential multiplicative cooling (emc, Eq. 12), linear multiplicative cooling (li-mc, Eq. 13) and logarithmic multiplicative cooling (log-mc, Eq. 14), with initial temperature.
Figure 7: Reconstruction errors (after 30 epochs) for different number of hidden nodes. (a) Reconstruction error for object nodes. (b) Reconstruction error for relation nodes and affordance nodes.

As shown in Figure 8, although annealing schedules decreases reconstruction errors all types of nodes, differences between them are not significant.

In summary, our analysis suggests that COSMO with one hidden layer, with 400 hidden nodes (although, as shown in Figure 7, 800 or more hidden nodes provide better performance, the performance gain is insignificant compared to the computational overload) and emc annealing performs best and therefore,
in the rest of the paper, we adopted these settings for COSMO. For RBM and GBM, we used the same number of hidden nodes as COSMO.

Figure 8: Reconstruction errors (after 30 epochs) for different annealing schedules with initial temperature 4.0. (a) Reconstruction error for object nodes. (b) Reconstruction error for relation and affordance nodes.

5.5. Comparison Measures

For evaluating the performance of the methods, we use precision, recall and F-measure which are defined as follows:

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{15}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{16}
\]

\[
\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}
\]
where TP, FP and FN stand for the number of true positives, false positives and false negatives, respectively. Definitions of TP, FP and FN are task-dependent, and therefore, they are defined for each task separately.

5.6. Task 1: Spatial Relation Estimation

Being generative, COSMO can estimate relations in the scene given the objects in the scene. Contextual information that arises from active objects, regardless of spatial relation and affordance nodes, allows the model to determine which spatial relations should be active according to the context.

For testing, initially, the model sees the environment in a “bag of objects” sense by clamping only objects to the visible nodes and relation nodes are set to zero. Next, the hidden nodes (i.e., context) are sampled using object nodes only. Then, the spatial relation nodes are sampled from objects, affordances and the context. This procedure is summarized in Algorithm 2.

\begin{algorithm}
\caption{Algorithm used for the relation estimation task (Task 1).}
\begin{algorithmic}[1]
\STATE \textbf{Input:} A scene $s$; the number of Gibbs steps, $k$.
\STATE \textbf{Output:} Relation node activations, $r$.
\STATE $r \leftarrow 0$. \Comment{Set relation node activations to 0.}
\FOR{$k$ sampling steps} \Comment{Clamp objects and affordances.}
\STATE $o \leftarrow s_o$, $a \leftarrow s_a$
\STATE Sample hidden nodes $h$ using Eq. \[8\]
\STATE Sample relation nodes $r$ using Eq. \[9\]
\ENDFOR
\end{algorithmic}
\end{algorithm}

For this task, we define True Positive (TP) as the number of spatial relation nodes which are active in both the input scene ($s$) and the reconstructed scene ($s'$); True Negative (TN) as the number of spatial relation nodes which are both in-active in $s$ and $s'$; False Positive (FP) as the number of spatial relation nodes which are inactive in $s$ but active in $s'$; False Negative (FN) as the number of spatial relation nodes which are active in $s$ and in-active in $s'$. These are defined
formally as follows:

\[ TP = \left| \{ x : x \in G_r^+ \land x \in M_r^+ \} \right|, \tag{18} \]
\[ TN = \left| \{ x : x \in G_r^- \land x \in M_r^- \} \right|, \tag{19} \]
\[ FP = \left| \{ x : x \in G_r^- \land x \in M_r^+ \} \right|, \tag{20} \]
\[ FN = \left| \{ x : x \in G_r^+ \land x \in M_r^- \} \right|. \tag{21} \]

where, \( x \) is a relation node; \( G_r^+ \) and \( G_r^- \) are respectively sets of active and passive relation nodes in the sample; and \( M_r^+ \) and \( M_r^- \) are sets of active and passive relation nodes respectively at the end of model’s reconstruction.

Table 3 lists the performance of COSMO for this task and compares it against RBM, GBM, and RN. In comparison to the other models, we see that COSMO provides the best performance.

Moreover, we provide some visual examples in Figure 9, where we see that our model can discover spatial relations between objects, i.e., how to roughly place a set of objects together.

In some cases, naturally, the “bag of objects” approach may not provide enough contextual information in order model to predict ground truth spatial relationships in the test set. For instance, consider a scene consisting of plate, table, cabinet objects. In a kitchen with eating context, the plate can be on the table, whereas, in a kitchen without eating context, plate is likely to be in the cabinet. These cases can reduce testing accuracy for the estimated rela-
Table 3: Task 1 (Spatial Relation Estimation) performances.

|        | Precision | Recall | F1-measure |
|--------|-----------|--------|------------|
| COSMO  | 0.1511    | 0.3112 | 0.2034     |
| GBM    | 0.1559    | 0.1125 | 0.1307     |
| RBM    | 0.0043    | 0.0132 | 0.0066     |
| RN     | 0.0166    | 0.0132 | 0.0147     |

tion between objects like *plate*. However, given such examples during training, COSMO is able to capture the probability of all these cases and therefore handle scene modeling tasks in such settings accordingly.

5.7. Task 2: What is missing in the scene?

In this task, COSMO predicts missing objects in the scene according to the current context. The model is provided “partially observed scenes” where some of the objects are removed randomly for testing.

Firstly, observed objects, spatial relations and affordances are clamped to the visible units, then the model is relaxed to find hidden node activations (i.e. the context of the scene). Finally, by using visible (scene description) and hidden (context) node activations, the network tries to find the missing objects in the scene as outlined in Algorithm 5.

For this task, we define TP as the number of object nodes that are activated correctly according to ground truth sample; FP as the number of object nodes that the model activates but it should be deactivated according to ground truth; TN as the number of object nodes that are deactivated correctly according to ground truth and FN as number of objects that the model deactivates yet should be activated according to ground truth. We can formally define these as follows:

\[
TP = |\{ x : x \in G_o^+ \land x \in M_o^+ \} |, \\
TN = |\{ x : x \in G_o^- \land x \in M_o^- \} |, \\
FP = |\{ x : x \in G_o^- \land x \in M_o^+ \} |, \\
FN = |\{ x : x \in G_o^+ \land x \in M_o^- \} |,
\]
where \( x \) is an object node; \( G^+_o \) and \( G^-_o \) are the sets of active and passive object nodes respectively in ground truth sample; and \( M^+_o \) and \( M^-_o \) are sets of active and passive object nodes respectively at the end of model’s reconstruction.

As shown in Table 4, our model performs better than RBM, GBM and RN. See also Figure 10, which shows some visual examples for most likely objects found for a target position in the scene.

**Algorithm 3** The algorithm for finding missing objects (Task 2).

1: **Input:** A scene, \( s \); the number of Gibbs steps, \( k \).
2: **Output:** Initially in-active object nodes in \( s \), \( o' \).
3: 
4: for \( k \) sampling steps do
5: \( o \leftarrow s_o; r \leftarrow s_r; a \leftarrow s_a \) \( \triangleright \) Clamp input scene.
6: Sample hidden nodes \( h \) using Eq. 8.
7: Sample in-active object nodes \( o' \) using Eq. 7.

---

Figure 10: Some examples illustrating the performance of COSMO on finding a missing object in a scene (Task 2).

5.8. **Task 3: What is extra in the scene?**

In this task, COSMO predicts objects that are out of context in the scene. For this purpose, objects are randomly selected and added to the original scene for testing.
Table 4: Task 2 (finding missing objects) performances.

|       | Precision | Recall | F1-measure |
|-------|-----------|--------|------------|
| COSMO | 0.9387    | 0.0527 | 0.0998     |
| GBM   | 0.8260    | 0.0415 | 0.0790     |
| RBM   | 0.7250    | 0.0301 | 0.0578     |
| RN    | 0.8000    | 0.0212 | 0.0414     |

Firstly, observed objects, spatial relations and affordances are clamped to the visible units, then the model is relaxed to find hidden node activations (i.e. the context of the scene). Finally, by using visible (scene description) and hidden (context) node activations, the network tries to remove objects that are out of context in the scene as outlined in Algorithm 4.

For this task, we use the TP, TN, FP and FN as defined in Equation 22.

As shown in Table 5, our model performs better than RBM, GBM and RN for finding extra objects in the scene. See also Figure 11 which shows some visual examples for finding the object that is out of context in the scene.

Table 5: Task 3 (finding extra objects) performances.

|       | Precision | Recall | F1-measure |
|-------|-----------|--------|------------|
| COSMO | 0.9183    | 0.0482 | 0.0917     |
| GBM   | 0.8113    | 0.0479 | 0.0865     |
| RBM   | 0.7826    | 0.0382 | 0.0729     |
| RN    | 0.7368    | 0.0297 | 0.0572     |

Algorithm 4 The algorithm for finding the out-of-context object (Task 3).

1: **Input:** A scene, $s$; the number of Gibbs steps, $k$.

2: **Output:** Initially active object nodes in $s$, $o'$.

3: 

4: for $k$ sampling steps do

5:  $o \leftarrow s_o$; $r \leftarrow s_r$; $a \leftarrow s_a$  

6:  Sample hidden nodes $h$ using Eq. 8

7:  Sample active object nodes $o'$ using Eq. 7

\[25\]
5.9. Task 4: Affordance Prediction

Affordances of objects may differ for different subjects in different contexts. Therefore, agents should be aware of the context that they are in in order to reason about the affordances of objects. We show that COSMO can allow agents to determine affordances of objects using the current context.

For this task, firstly, objects and relations are clamped to the visible nodes, and the hidden nodes (i.e. context) are sampled. Then, affordance nodes are sampled using hidden nodes (context), objects and relations (current scene), as illustrated in Algorithm 5.

Algorithm 5 Algorithm for the affordance prediction task (Task 4).

1: Input: A scene, $s$; the number of Gibbs steps, $k$.
2: Output: Affordance node activations, $a$.
3: 
4: $a \leftarrow 0$.  \hspace{1cm} \triangleright$Set affordance nodes to 0.$
5: for $k$ sampling steps do
6: \hspace{1cm} $o \leftarrow s_o$, $r \leftarrow s_r$.  \hspace{1cm} \triangleright$Clamp objects and relations.$
7: \hspace{1cm} \text{Sample hidden nodes $h$ using Eq. 8}$
8: \hspace{1cm} \text{Sample affordance nodes $a$ using Eq. 10}$

For this task, we define TP as the number of affordance nodes that are
activated correctly according to the ground truth sample; FP as the number of affordance nodes that the model activates but should be deactivated according to the ground truth; TN as the number of affordance nodes that are deactivated correctly according to ground truth, and FN as the number of affordance nodes that the model deactivates yet should be activated according to the ground truth. We defined them formally as follows:

\[ TP = \left| \{ x : x \in G_a^+ \land x \in M_a^+ \} \right|, \]

\[ TN = \left| \{ x : x \in G_a^- \land x \in M_a^- \} \right|, \]

\[ FP = \left| \{ x : x \in G_a^- \land x \in M_a^+ \} \right|, \]

\[ FN = \left| \{ x : x \in G_a^+ \land x \in M_a^- \} \right|, \]

where, \( x \) is an affordance node; \( G_a^+ \) and \( G_a^- \) are sets of active and passive affordance nodes respectively in the ground truth sample; and \( M_a^+ \) and \( M_a^- \) are the sets of active and passive affordance nodes respectively at the end of model’s reconstruction.

As shown in Table 6, our model performs better than RBM, GBM and RN. See also Figure 12(a), which shows some visual examples for affordance prediction for different objects.

Table 6: Task 4 (affordance prediction) performances.

|       | Precision | Recall | F1-measure |
|-------|-----------|--------|------------|
| COSMO | 0.2039    | 0.3129 | 0.2469     |
| GBM   | 0.1372    | 0.1068 | 0.1201     |
| RBM   | 0.0769    | 0.0076 | 0.0138     |
| RN    | 0.0125    | 0.0091 | 0.0105     |

5.10. Task 5: Objects affording an action

Being generative, COSMO allows reasoning about object affordances in various ways. In this task, we evaluate the methods on finding objects that afford a certain action. For this end, some of the object nodes, which are the object
part of an affordance-triplet, and corresponding affordance nodes are deactivated. Then, the model samples hidden nodes using the partially observed scene. In the reconstruction phase, the model samples deactivated objects and affordance nodes that includes these objects using context and observed scene. This is formalized in Algorithm 6.

For this task, we use same formal definitions of TP, FP, TN and FN in Equation 23. However, in this task, $G_a^+, G_a^-, M_a^+$ and $M_a^-$ include affordance nodes that correspond to a specific action and subject instead of all affordance nodes.

Table 7 lists the performance of the methods for finding the objects affording a certain action. We see a significant difference between the performance of COSMO and those of GBM, RBM and RN. See also Figure 12(b), which shows some visual examples for predicting the object that affording specific action.

**Algorithm 6** The algorithm for finding objects that afford a given action (Task 5).

1. **Input:** A scene, $s$; an action, $act$; the subject of action, $subj$; the number of Gibbs steps, $k$.
2. **Output:** Active affordance nodes, $a_{ia,isa_o}$.
3. $ia = \text{index of } act \text{ in affordance vocabulary.}$
4. $io = \text{index of } subj \text{ in object vocabulary.}$
5. **for** $k$ sampling steps **do**
6. $o \leftarrow s_o$ \hspace{1cm} $\triangleright$ Clamp objects to the visible nodes.
7. $r \leftarrow s_r$ \hspace{1cm} $\triangleright$ Clamp relations to the visible nodes.
8. $a \leftarrow s_a$ \hspace{1cm} $\triangleright$ Clamp affordances to the visible nodes.
9. Sample hidden nodes $h$ using Eq. 8.
10. Sample affordance node $a_{ia,isa_o}$ using Eq. 7.

5.11. **Task 6:** Who is the actor for this task?

Robots should also be able to reason about the possible actors (subjects) of a given action or a task. Context plays a critical role here since it can modulate
Table 7: Task 5 (finding objects affording a given action) performances.

|         | Precision | Recall | F1-measure |
|---------|-----------|--------|------------|
| COSMO   | 0.3170    | 0.4482 | 0.3714     |
| GBM     | 0.2537    | 0.0739 | 0.1144     |
| RBM     | 0.0740    | 0.0869 | 0.0800     |
| RN      | 0.0157    | 0.0689 | 0.0256     |

Predicted affordances in the scene:
- Human carrying suitcase
- Human holding suitcase

(a) What can human hold?
- Human 0.23
- Pan 0.92
- Oven 0.12

(b) Figure 12: Some examples illustrating (a) the performance of COSMO on affordance prediction (Task 4) and (b) finding objects that affording specific action (Task 5).

In this task, we evaluate performances on finding proper subjects (actors) for a given action.

In this task, we evaluate performances on finding proper subjects (actors) for a certain action with a specific object. For this end, some of the object nodes, which are the subject part of an affordance-triplet, and affordance nodes are deactivated. Then, the model samples hidden nodes using the partially observed scene. In the reconstruction phase, the model samples deactivated objects and affordance nodes that have proper subject for given action by using context and observed scene. This is formalized in Algorithm 7.

For this task, we use same formal definitions of TP, FP, TN and FN in Equation 23. However, in this task, $G^+_a$, $G^-_a$, $M^+_a$ and $M^-_a$ include affordance nodes that correspond to a specific action and object instead of all affordance nodes.

In Table 8 the performances of the methods are listed. We see that GBM performs better in terms of precision whereas COSMO yields a much better recall performance, leading to an overall better performance in terms of the
Algorithm 7 The algorithm for finding the subject for a given action (Task 6).
1: **Input:** A scene, \( s \); an action, \( act \); the object of action, \( obj \); the number of Gibbs steps, \( k \).
2: **Output:** Active affordance nodes, \( a_{i_a, i_o} \).
3: \( i_a \) = index of \( act \) in affordance vocabulary.
4: \( i_o \) = index of \( obj \) in object vocabulary.
5: **for** \( k \) sampling steps **do**
6: \( o \leftarrow s_o \) \hspace{1cm} \triangleright \text{Clamp objects to the visible nodes.}
7: \( r \leftarrow s_r \) \hspace{1cm} \triangleright \text{Clamp relations to the visible nodes.}
8: \( a \leftarrow s_a \) \hspace{1cm} \triangleright \text{Clamp affordances to the visible nodes.}
9: Sample hidden nodes \( h \) using Eq. 8.
10: Sample affordance node \( a_{i_a, i_o} \) using Eq. 7.

Table 8: Task 6 (What is the actor of the affordance?) performances.

| Model  | Precision | Recall | F1-measure |
|--------|-----------|--------|------------|
| COSMO  | 0.3055    | 0.4782 | 0.3728     |
| GBM    | 0.3333    | 0.0689 | 0.1142     |
| RBM    | 0.0539    | 0.0586 | 0.0561     |
| RN     | 0.0312    | 0.1739 | 0.0529     |

5.12. **Task 7: Improving Object Detection**

In this task, we test whether we can use COSMO to rectify wrong detections and find missing detections made by object detectors. For this purpose, we used three state-of-the-art three object detection networks (namely, RetinaNet 53, Faster R-CNN 54, and Mask R-CNN 55) with the ResNet-101-FPN 56 backbone model trained on the COCO dataset 57.

For this task, we first run the deep object detector on the input image. Then, we provide the detected objects to COSMO, and relax the network to see how COSMO updates the object nodes. We calculate average precision over 100
randomly selected images and compare the performance of the deep detectors before and after applying COSMO.

As shown in Table 9, COSMO significantly improves the detection performance of the deep networks. Looking at the visual example provided in Figure 13, we observe that COSMO can correct the mistakes made by the object detectors, and suggest objects that were missed by the detectors.

![Object Detector result: Microwave 0.92 Bottle 0.82 Refrigerator 0.88 Remote 0.07 COSMO result: Microwave 0.92 Bottle 0.82 Refrigerator 0.88 Remote 0.07](image)

Table 9: Task 7: Improving object detections with COSMO. The average precision for different object detectors with and without COSMO are listed.

| Object Detector       | w/o COSMO | w COSMO |
|-----------------------|-----------|---------|
| RetinaNet [53]        | 0.4964    | 0.6966  |
| Faster R-CNN [54]     | 0.4388    | 0.6752  |
| Mask R-CNN [55]       | 0.4273    | 0.6648  |
Figure 14: An example illustrating scene generation capability of COSMO (Task 8). (a) When a context (hidden node) is activated, (b) active nodes in the sampled visible nodes define a scene for the context. In (b), the “selected” objects are placed in the scene based on the predicted spatial relations.

5.13. Task 8: Random scene generation

In this task, we demonstrate how we can use another generative ability of COSMO: we can select a hidden node (or more of them, leaving the other hidden neurons randomly initialized or set to zero), and sample visible nodes (including relations and affordances) that describe a scene. Figure 14 shows a visual example.

5.14. Task 9: Experiments on a Real Robot

In this task, we evaluate COSMO on Nao and illustrate how the tasks 1-8 conducted in this section can be useful for a robot. For this purpose, Nao uses Mask R-CNN to detect objects in the scene, and COSMO is initialized with these detections (only object nodes are clamped with the detected objects, the other visible nodes (relations and affordances) are estimated after sampling the hidden nodes). See Figure 15 for a snapshot.

Once COSMO is relaxed, Nao can reason about objects, relations, affordances, missing objects or out-of-context objects in the scene. An interactive experiment has been conducted with Nao where Nao answers questions about
Figure 15: A snapshot of an experiment performed with Nao (Task 9). Nao uses Mask R-CNN to detect objects in the scene, and COSMO is initialized with these detections. Once this has been performed, Nao can reason about relations, affordances, missing objects or out-of-context objects in the scene. See the accompanying video (also provided at https://bozcani.github.io/COSMO) for the experiments.

6. Conclusion

In this paper, we proposed a novel method (COSMO) for contextualized scene modeling. For this purpose, we extended Boltzmann Machines (BMs) to include spatial relations and affordances via tri-way edges in the model. For integrating spatial relations and affordances into the model, we introduced shared nodes into BMs, allowing the concept of relations and affordances to be shared among different objects pairs. We evaluated and compared our model on several tasks on a real dataset and a real robot platform.

On several challenging tasks, we demonstrated that our model is very suitable for scene modeling purposes with its generative and explicit nature. Being generative, we showed that a single COSMO model allows reasoning about many
aspects of the scene given any partial information. On these tasks, COSMO performed consistently better in comparison to the baseline methods (general BMs and restricted BMs) and relational networks [50].

Acknowledgment

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) through project called “Context in Robots” (project no 215E133). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

[1] W. Yeh, L. W. Barsalou, The situated nature of concepts, The American journal of psychology (2006) 349–384.

[2] L. W. Barsalou, Simulation, situated conceptualization, and prediction, Philosophical Transactions of the Royal Society B: Biological Sciences 364 (1521) (2009) 1281–1289.

[3] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, A learning algorithm for boltzmann machines, Cognitive science 9 (1) (1985) 147–169.

[4] T. J. Sejnowski, Higher-order boltzmann machines, in: AIP Conference Proceedings, Vol. 151, AIP, 1986, pp. 398–403.

[5] D. Joho, G. D. Tipaldi, N. Engelhard, C. Stachniss, W. Burgard, Nonparametric bayesian models for unsupervised scene analysis and reconstruction, Robotics (2013) 161.

[6] A. Anand, H. S. Koppula, T. Joachims, A. Saxena, Contextually guided semantic labeling and search for three-dimensional point clouds, The International Journal of Robotics Research 32 (1) (2013) 19–34.
[7] S. Blumenthal, H. Bruyninckx, Towards a domain specific language for a scene graph based robotic world model, arXiv preprint arXiv:1408.0200.

[8] H. Çelikkanat, G. Orhan, S. Kalkan, A probabilistic concept web on a humanoid robot, IEEE Transactions on Autonomous Mental Development 7 (2) (2015) 92–106.

[9] F. Mastrogiovanni, A. Scalmato, A. Sgorbissa, R. Zaccaria, Robots and intelligent environments: Knowledge representation and distributed context assessment, Automatika 52 (3) (2011) 256–268.

[10] H. Çelikkanat, E. Şahin, S. Kalkan, Integrating spatial concepts into a probabilistic concept web, in: IEEE International Conference on Advanced Robotics (ICAR), 2015.

[11] A. Pronobis, P. Jensfelt, Large-scale semantic mapping and reasoning with heterogeneous modalities, in: IEEE International Conference on Robotics and Automation (ICRA), 2012.

[12] W. Hwang, J. Park, H. Suh, H. Kim, I. H. Suh, Ontology-based framework of robot context modeling and reasoning for object recognition, in: Int. Conf. on Fuzzy Systems and Knowledge Discovery, 2006.

[13] X. Li, J.-F. Martínez, G. Rubio, D. Gómez, Context reasoning in underwater robots using mebn, arXiv preprint arXiv:1706.07204.

[14] X. Wang, E. Grimson, Spatial latent dirichlet allocation, in: Advances in Neural Information Processing Systems (NIPS), 2008, pp. 1577–1584.

[15] J. Philbin, J. Sivic, A. Zisserman, Geometric LDA: A generative model for particular object discovery, in: British Machine Vision Conference (BMVC), 2008.

[16] D. Lin, S. Fidler, R. Urtasun, Holistic scene understanding for 3d object detection with rgbd cameras, in: IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1417–1424.
[17] H. Celikkanat, G. Orhan, N. Pugeault, F. Guerin, E. Şahin, S. Kalkan, Learning context on a humanoid robot using incremental latent dirichlet allocation, IEEE Transactions on Cognitive and Developmental Systems 8 (1) (2016) 42–59.

[18] M. Tenorth, M. Beetz, Knowrob – knowledge processing for autonomous personal robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2009.

[19] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, H. S. Koppula, Robobrain: Large-scale knowledge engine for robots, arXiv preprint arXiv:1412.0691.

[20] Y. Sheikh, M. Shah, Bayesian modeling of dynamic scenes for object detection, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 27 (11) (2005) 1778–1792.

[21] E. Stopp, K.-P. Gapp, G. Herzog, T. Laengle, T. C. Lueth, Utilizing spatial relations for natural language access to an autonomous mobile robot, in: Annual Conference on Artificial Intelligence, Vol. 861, Springer Science & Business Media, 1994, p. 39.

[22] P. Meissner, R. Reckling, R. Jakel, S. R. Schmidt-Rohr, R. Dillmann, Recognizing scenes with hierarchical implicit shape models based on spatial object relations for programming by demonstration, in: IEEE International Conference on Advanced Robotics (ICAR), 2013.

[23] S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, T. Darrell, Grounding spatial relations for human-robot interaction, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013.

[24] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, R. Girshick, Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, arXiv preprint arXiv:1612.06890.
[25] J. J. Gibson, The Ecological Approach to visual perception, Lawrence Erlbaum Associates, 1986.

[26] E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, G. Üçoluk, To afford or not to afford: A new formalization of affordances toward affordance-based robot control, Adaptive Behavior 15 (4) (2007) 447–472.

[27] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, L. De Raedt, Learning relational affordance models for robots in multi-object manipulation tasks, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2012, pp. 4373–4378.

[28] E. Uğur, M. R. Doğar, M. Cakmak, E. Sahin, The learning and use of traversability affordance using range images on a mobile robot, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2007, pp. 1721–1726.

[29] M. Lopes, F. S. Melo, L. Montesano, Affordance-based imitation learning in robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007, pp. 1015–1021.

[30] K. F. Uyanik, Y. Calskan, A. K. Bozcuoglu, O. Yuruten, S. Kalkan, E. Sahin, Learning social affordances and using them for planning, in: Proceedings of the Annual Meeting of the Cognitive Science Society, Vol. 35, 2013.

[31] S. Kalkan, N. Dag, O. Yürüten, A. M. Borghi, E. Şahin, Verb concepts from affordances, Interaction Studies 15 (1) (2014) 1–37.

[32] I. Atıl, N. Dag, S. Kalkan, E. Sahin, Affordances and emergence of concepts, in: 10th International Conference on Epigenetic Robotics, 2010.

[33] P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Uğur, J. Piater, Computational models of affordance in robotics: a taxonomy and systematic classification, Adaptive Behavior 25 (5) (2017) 235–271.
[34] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater,
J. Santos-Victor, Affordances in psychology, neuroscience and robotics: a
survey, IEEE Transactions on Cognitive and Developmental Systems.

[35] H. S. Koppula, R. Gupta, A. Saxena, Learning human activities and ob-
ject affordances from rgb-d videos, The International Journal of Robotics
Research 32 (8) (2013) 951–970.

[36] L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor, Modeling affor-
dances using bayesian networks, in: IEEE/RSJ International Conference
on Intelligent Robots and Systems (IROS), 2007, pp. 4102–4107.

[37] L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor, Learning object
affordances: From sensory–motor coordination to imitation, IEEE Trans-
actions on Robotics 24 (1) (2008) 15–26.

[38] A. Bouliaras, O. Kroemer, J. Peters, Learning robot grasping from 3-d
images with markov random fields, in: IEEE/RSJ International Conference
on Intelligent Robots and Systems (IROS), IEEE, 2011, pp. 1548–1553.

[39] A. Nguyen, D. Kanoulas, D. G. Caldwell, N. G. Tsagarakis, Detecting
object affordances with convolutional neural networks, in: IEEE/RSJ In-
ternational Conference on Intelligent Robots and Systems (IROS), IEEE,
2016, pp. 2765–2770.

[40] T.-T. Do, A. Nguyen, I. Reid, D. G. Caldwell, N. G. Tsagarakis, Affor-
dancenet: An end-to-end deep learning approach for object affordance de-
tection, arXiv preprint arXiv:1709.07326.

[41] M. Kokie, J. A. Stork, J. A. Haustein, D. Kragic, Affordance detection
for task-specific grasping using deep learning, in: IEEE-RAS 17th Inter-
national Conference on Humanoid Robotics (Humanoids), IEEE, 2017, pp.
91–98.

[42] R. Salakhutdinov, G. Hinton, Deep boltzmann machines, in: Artificial In-
telligence and Statistics, 2009, pp. 448–455.
[43] R. Salakhutdinov, A. Mnih, G. Hinton, Restricted boltzmann machines for collaborative filtering, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 791–798.

[44] I. Bozcan, Y. Oymak, İ. Z. Alemdar, S. Kalkan, What is (missing or wrong) in the scene? a hybrid deep boltzmann machine for contextualized scene modeling, Accepted for IEEE International Conference on Robotics and Automation (ICRA) 2018; arXiv preprint arXiv:1710.05664.

[45] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, et al., Visual genome: Connecting language and vision using crowdsourced dense image annotations, International Journal of Computer Vision 123 (1) (2017) 32–73.

[46] R. M. Neal, Connectionist learning of belief networks, Artificial intelligence 56 (1) (1992) 71–113.

[47] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (6) (1984) 721–741.

[48] D. M. Blei, A. Kucukelbir, J. D. McAuliffe, Variational inference: A review for statisticians, Journal of the American Statistical Association (just-accepted).

[49] S. Song, S. P. Lichtenberg, J. Xiao, Sun rgb-d: A rgb-d scene understanding benchmark suite, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 567–576.

[50] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, T. Lillicrap, A simple neural network module for relational reasoning, in: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 4974–4983.
[51] Y. Nourani, B. Andresen, A comparison of simulated annealing cooling strategies, Journal of Physics A: Mathematical and General 31 (41) (1998) 8373.

[52] M. Tucker, R. Ellis, On the relations between seen objects and components of potential actions., Journal of Experimental Psychology: Human perception and performance 24 (3) (1998) 830.

[53] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, arXiv preprint arXiv:1708.02002.

[54] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 91–99.

[55] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 2980–2988.

[56] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision (ECCV), Springer, 2014, pp. 740–755.