International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Learning Object Affordances from Sensory-Motor Interaction via Bayesian Networks with Auto-Encoder Features

Mete Tuluhan Akbulut[*1], Emre Ugur[2]

Submitted: 04.11.2019 Accepted: 19.05.2020

Abstract: In this paper, we study learning relationships between objects, actions, and effects. "Affordance" is an ecological psychology concept, which addresses how humans learn these relationships and which is also studied in cognitive robotics to transfer the same ability to robots. Our model is built on top of two existing models in this field and uses their strengths to introduce a novel system, where an anthropomorphic robot observes its environment and changes in that environment after executing pre-learned actions. Robot transfers these observations to object and effect properties in the same space and object affordances are learned using Bayesian Networks. The dimensions of features are decreased through autoencoders to achieve a compact network. Usage of a probabilistic model helps our system to deal with missing information or to make predictions for object properties and actions along with effect properties. We illustrate the advantages of our model by comparing it with the two aforementioned models.

Keywords: *affordance, cognitive robotics, developmental robotics, perception, learning*

1. Introduction

Humans solve many tasks in their daily life without recognizing how complex they are. It is more obvious to researchers in artificial intelligence (AI) and robotics because computers can do better than humans in games [1] but progress of AI is much slower when it comes to addressing real-world problems [2].

To achieve human-level performance in real-world tasks, robots should be able to understand the characteristics of the environment and to estimate how their actions change the environment. In this article, we approach this challenge from a developmental perspective. Before making plans to achieve goals, growing infants first make sense of incoming stimuli by learning how they can change the environment. Babies manipulate the objects differently according to their properties while they are 6 months old [3]. After 6 months, they start to make sense of not only objects and surfaces but also the relations between them [4]. As they grow up, the perceived properties of the environment change in relation to their abilities. Crawling children find both rigid and non-rigid surfaces navigable, whereas walking children learn that non-rigid objects are not easily navigable [5].

This phase is explained with the phenomenon "affordance" in developmental psychology [6]. It is defined by J.J. Gibson as directly perceived "meanings" of surroundings in an organism's environment and provided action possibilities. For example, detecting the affordance of a bottle is analogous with the question: "Which actions can be taken using this bottle?". However, it is important to note that this relation does not necessarily require object recognition [7]. The actions are linked to the object's features directly.

In this work, we model affordances for robots on top of two existing works on affordances. The first approach [29,34,39] used Support Vector Machines to learn affordances. Support Vector Machine (SVM) is a strong method to model affordances, but it is a deterministic method that allows only unidirectional prediction.

The second one [25,31,32,33] used Bayesian Networks to learn affordances. Bayesian Network (BN) is a graph that represents relations between variables of data in terms of probability [8]. It is a directed acyclic graph and each directed edge defines a dependency, i.e. conditional probability, of the incoming node on the outcoming node. Bayesian Networks do not try to fit a function like standard supervised learning algorithms, thus are less prone to overfitting. When some input information is missing, Bayesian Networks can make predictions where standard supervised algorithms do not perform well. This representation is very useful for the affordances because the example question above can be rewritten as: "Which action is more probable to produce the desired effect for this bottle?". The network visualizes these dependencies and helps the computation of probabilities. Unknown probabilities can be inferred omnidirectional according to Bayes' theorem. The nodes of the Bayesian Network in [25] were object categories, effect categories, and action categories. However, the object and effect types were not learned directly from data. They were predefined and encoded as different nodes in BN. For example, color, shape, and size were defined for the objects; and object velocity, contact time, and object-hand distance were defined for the effects. Thus, the number of nodes of the Bayesian Network increased with the number of features, and it became harder to make an inference. Raghvendra and Inamura extended Bayesian Networks for learning tool affordances by including tool related nodes [33]. Saponaro et al. extended this framework to learning affordances along with the verbal descriptions [31]. Then, Hidden Markov Models with Gaussian Mixture Models were used to interpret and to describe other agents' actions. Andries et al. [32] formalized affordances with Bayesian Networks emphasizing the equivalence classes in affordances. However, effect and object types were predefined in these studies.

In our work, we use Bayesian Networks instead of SVMs to learn

[1] *Computer Eng., Bogazici University, Istanbul–34342, TURKEY*
ORCID ID: 0000-0001-5822-1031
[2] *Computer Eng., Bogazici University, Istanbul–34342, TURKEY*
ORCID ID: 0000-0001-9597-2731
** Corresponding Author: Email: tuluhan.akbulut@boun.edu.tr*

the affordances and our model learns the categories of the object and the effect directly from data. Furthermore, instead of pre-grouping data as in [25], we use neural networks to decrease dimensions of data in order to obtain a single object type and a single effect type. In particular, we use an autoencoder that learns to produce outputs as close as to the inputs [9], forcing the input through a bottleneck layer composed of a small number of neurons. The activations at the bottleneck layer are used as the low-dimensional representation in a latent space, through which K-means clustering is applied to find data- driven categories. As a result, our Bayesian Network is formed using only 3 nodes for object, action, and effect; which in turn allows efficient inference. Additionally, clustering is not done based on only linear relations of features as in [25] and [29], because auto-encoders extract nonlinear relationships in data as well.

Our contribution can be summarized as follows: (1) extension of our previous work with a probabilistic approach, (2) use of autoencoders to avoid pre-engineered Bayesian Network nodes and keeping network size at a minimum to allow efficient inference, and (3) qualitative and quantitative comparison of discriminative and generative approaches, i.e. SVM-based and Bayesian Network based affordance learning methods.

2. Related Work

Using concepts in developmental psychology in robotics has become popular in recent years [10, 11, 12, 13]. In these studies robots explore and learn about their environment with minimal help from experts. In this section, we review studies in this field and emphasize our contribution.

The prominent studies that use motor activity to explore the environment and learn affordances include Metta and Fitzpatrick [14], Fitzpatrick et al. [15] and Stoytchev [16]. As an example, Stoytchev [16] studied tool affordances by discovering tool-behavior pairs that give the desired effect. However, in these studies the learned affordances could not be generalized on novel objects/tools, because the learned relations between visual features and effects of tools were not addressed.

The relations between visual features and effects were addressed by Ugur and Sahin [17], Erdemir et al. [18], Fritz et al. [19], where associations between visual features of objects and their affordances were learned. In these studies, the affordance categories were defined by the programmer and supervised learning was implemented. So, the robot did not explore the environment and learned affordances, rather it predicted predefined effects. Sinapov and Stoytchev [20], Griffith et al. [21], Cos-Aguilera et al. [22] studied self-discovery of effects and affordances. In these studies, effect categories were learned through unsupervised clustering in effect space and these categories were mapped to objects so that the robot was able to choose actions that would result in desired effects.

The studies that we discussed were all deterministic and could learn and reason about one-directional mappings, i.e. they could predict the effects given object features and robot actions, but they could not predict the object features given robot actions and desired effects. Demiris and Dearden [23], Hart et. al. [24], Raghvendra and Inamura [33], Saponaro et al. [31], Andries et al. [32], and Montesano et. al. (2008) [25] used probabilistic networks to learn relations between actions, objects, and effects. Probabilistic networks allow learning multi-directional relations. For example in Montesano et al. [25], Bayesian Networks encoded these relations and robot was able to predict the object categories given that effect categories and actions were known, or it could also predict effect categories given that object categories and actions were known, or it could also estimate the required action to obtain a given effect category for a given object. On the other hand, object features and different effects were modeled with different nodes and as the number of nodes increases, the number of possible network structures increases exponentially. Therefore, it would be impractical to explore all possible graphs. They proposed different methods to make inference such as Markov chain Monte Carlo (MCMC) method or the maximum likelihood solution provided by the K2 algorithm.

Recently, deep learning approaches gained popularity for learning locomotion and manipulation affordances. Do et. al. [38] used Convolutional Neural Networks (CNN) to detect objects and affordances from raw RGB images. Seker et al. [35] used CNN and Long Short-Term Memory (LSTM) to predict push affordances and low-level outcome trajectories of levered-up objects from their depth image. Chu et al. [36] realized a deep learning framework that predicts the affordances of object parts for robotic manipulation through segmenting affordance maps by jointly detecting and localizing candidate regions within an image. Li et al. [37] exploited deep residual U-Nets for learning grasp affordances. While deep learning affordance classifiers acquire high performances in the corresponding datasets, they generally require large numbers of training samples, which is challenging and time consuming to obtain from robotics interactions.

None of the methods above are fit for complex planning because of the effect-representation they have. It is not possible to plan one step further with these representations. In [26, 27, 28] the robots could make multi-step predictions by linking actions according to logical precondition and postcondition predicates. For example, Wörtgötter et al. [27] defined conditions as binary functions of sensor readings and then the robot learned to remodel these conditions in the form of pre-conditions and effects with the help of a human expert. Ugur et al. [29] showed that complex planning can be achieved if effects and objects were encoded in the same feature space. By adding the object features of the current state to the predicted effect features, the robot could predict the next perceptual state, therefore it was able to achieve multi-step planning. In [29], Support Vector Machines were used in predicting effects from objects and actions. This made the model deterministic and relations between objects and effects unidirectional.

In our study, we combine the Bayesian Network idea with the SVM learning approach to make the model probabilistic and relations bidirectional. Directly applying Bayesian Networks to the model assumes human expert domain knowledge, because Bayesian Networks require discretization. On the other hand, the discretization performed by humans is not scalable to life-long learning settings and might not correspond to the sensorimotor capabilities of the robot. Classic clustering algorithms to discretize the data do not work simply because of the aforementioned drawback of [25]. An increasing number of features result in exponential growth of the number of possible Bayesian Networks. We propose to transfer the object features into a latent space through autoencoders first and only then apply clustering algorithms so that we could have only one node for objects and one node for effects without any pre-engineering. Besides, in [25] and [29] X-means clustering algorithm was applied to data for categorization and clusters were formed based on Euclidean distances in feature space. However, since autoencoders extract nonlinear relationships in data, using them before categorization enables the model to cluster data based on nonlinear and complex relations.

3. The Approach

reduction, pixels at the edge of the objects are removed and Median and Gaussian filters with a 5x5 sized window are applied. In the end, feature vectors of each object are determined as follows.

Three categories of information are encoded in the feature vector of the object: Object visibility that encodes the existence of the object; object position that is represented by the six values along longitudinal, lateral and vertical axes and object shape that corresponds to the distribution of local surface normal vectors. The normal vector around each point is computed using positions of the two neighbors in the range image:

$$N_{r,c} = (d_{r-n,c} - d_{r,c}) \times (d_{r,c-n} - d_{r,c}) \quad (1)$$

Here, d represents the 3D position, n corresponds to neighbor pixel distance and it can be set to an arbitrary value. The normal vector is calculated in a spherical coordinate system, where the vector is represented with radius, analogous to the distance from the origin, polar angle, the angle between the normal vector and x-z plane, azimuthal angle, the angle between the projection of normal vector to the x-z plane and z-axis. After angles are calculated for every pixel, two histograms are formed for the 2 angles and they express 36 shape features.

For interaction, one lift action and three push actions are defined, and the robot is capable of performing these actions. The robot perceives the positions of objects by its camera and it is able to interact with the objects at different positions. For the three push actions, namely push-left, push-right and push forward, the robot hand is moved to the corresponding side of the object and push-action is done towards the center of the object. In order to lift the object, the robot puts its hand at the back-right diagonal of the object and moves it towards the object while closing the fingers. After grasping, the object is lifted vertically. The objects that the robot interacts with can be categorized as cubic, cylindrical and spherical. Different types of objects result in different types of effects after action execution. For example, pushing a cubic object makes its position change but pushing a spherical object makes it roll away from the table so that the camera cannot see it anymore. Also, the result of pushing a cylindrical object varies according to the side of the object that force is applied.

In the explained way, the robot collects object features via its physical simulator and interacts with the objects randomly. The final features of the object are also recorded after the action is applied. An effect vector is acquired by subtracting initial feature vector from the final feature vector:

$$f_{effect} = f_{end} - f_{initial} \quad (2)$$

Learning affordances require abstraction of observation. The model uses discrete random variables to achieve this abstraction. For example, object features can be represented with discrete random variables for shape, position such as $O = \{s_1, s_2, ..., p_1, p_2, ...\}$. However, many random variables cause trouble when constructing the Bayesian Network. So, dimension reduction for the object and the effect features is needed.

For that purpose, we use autoencoders. In Fig. 1, it can be seen that a neural network is designed to construct an output as close as possible to its input through a bottleneck layer. The activations in the bottleneck layer represent input features and the space that consists of these activations is called latent space. In our model, the first layer has the same size as the input feature vector. Then in every layer, the dimension size is reduced to its half until 3 nodes are enough to represent the data. More nodes may increase accuracy, but 3 nodes work fine in our case since our concern is to decrease the dimension number for clustering. After the bottleneck layer, in every layer, dimension size is doubled until we have a layer that has the same size as the input vector. Since dimensions of latent space are fewer than the original space, the possible BN network structure decreases. After the initial features are transferred into a latent space, the K-means clustering algorithm is applied to categorize the data. Effect features are also clustered with the K-means algorithm, which is an unsupervised algorithm that selects K number of centers and assigns the data points to the closest center to group the data. Afterwards, new centers of groups are determined, and grouping is done again. The algorithm stops when no point needs to change its cluster. In the end, the algorithm gives clusters, where data points are similar according to positions in defining space.

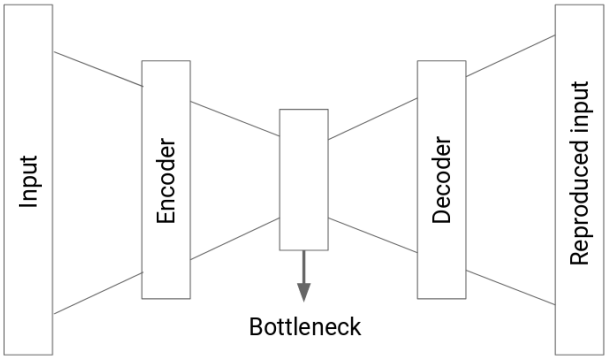

Fig. 1. Autoencoder schematic, the activations of the bottleneck layer are the representation of the input in latent space. Encoder decreases the input dimension in a way so that decoder can reconstruct the input.

After grouping object and effect features, all that is left is to encode the relationship between the actions, the object clusters and the effect clusters. In order to do this, Bayesian Networks are used.

Bayesian Network is a graph: G (V, E), whose edges specify conditional probabilities. Thus, it is a convenient way to illustrate and model affordances. It allows deducting probabilities for a node in condition with other nodes or the joint probabilities in terms of conditional probabilities:

$$p(V_1 | V_2) = p(V_1, V_2) / p(V_2) \quad (3)$$

$$p(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} p(V_i | parents(V_i)) \quad (4)$$

Fig. 2. shows a network with 3 nodes and 2 edges. BNs need discretization and that is why clustering takes place in the previous step.

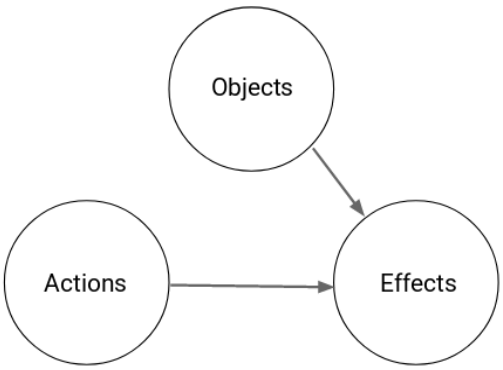

Fig. 2. Bayesian Network for affordances. Effects are conditioned on which action or object are selected. Objects and actions can be selected arbitrarily, so they are independent variables. However, some actions coupled with some objects are more likely to produce a desired effect. Therefore, actions and objects are not conditionally independent.

To form the network, the conditional probabilities of child nodes and the probability distribution of parent nodes should be calculated in the data:

$$p(a),p(e|o,a),p(e|o,\bar{a}),$$
$$p(o),p(e|\bar{o},a),p(e|\bar{o},\bar{a})$$

Here, "a,e,o" stand for a single action, object and effect, respectively and "$\bar{a},\bar{e},\bar{o}$" represents for actions, objects and effects alternative to "a,e,o". After the network is formed, it can infer probabilities for each node by following Bayes' rule:

$$p(a|o,e)=\frac{p(e|a,o)\times p(a)}{p(e|a,o)\times p(a)+p(e|\bar{a},o)\times p(\bar{a})} \quad (5)$$

$$p(o|a,e)=\frac{p(e|a,o)\times p(o)}{p(e|a,o)\times p(o)+p(e|\bar{o},a)\times p(\bar{o})} \quad (6)$$

It is also possible to infer the probabilities even in the case of incomplete data. Neural networks or other standard methods like SVM cannot handle incomplete data well but Bayesian Networks can provide good predictions with the formulas below:

$$p(e|o)=p(e|o,a)\times p(a)+p(e|o,\bar{a})\times p(\bar{a}) \quad (7)$$

$$p(e|a)=p(e|o,a)\times p(o)+p(e|\bar{o},a)\times p(\bar{o}) \quad (8)$$

$$p(a|e)=\frac{p(e|a)\times p(a)}{p(e)} \quad (9)$$

$$p(o|e)=\frac{p(e|o)\times p(o)}{p(e)} \quad (10)$$

4. Experiments

4.1. 4.1 Experimental Setup

The simulated robot interaction data obtained in [29] is used in this paper. The robotic system is composed of a five-fingered 16 DOF robot hand and a 7 DOF robot arm. The infrared range camera provides a 176x144 pixel array with 1 cm distance accuracy, and 0.23-degree angular resolution. The camera is able to produce a grayscale image of the scene and a confidence value for each pixel. 7 types of objects were used: There were 2 balls, 2 cylinders, 2 rectangular prisms with different sizes and 1 pitcher. Different shapes and sizes were selected to make robots learn related affordances.

The robot explored its environment in the simulator as follows (Fig. 3): The robot perceived its environment before and after interacting with one of the random objects placed in a random position with one of its four actions. Approximately 10500 interactions were performed for each action type and feature vectors of objects before and after an action execution were recorded. Both the initial and final feature vector were composed of 43 features, including 1 visibility, 6 position, and 36 shape-related features (Fig. 4). The effect feature vector was found by subtracting the initial feature vector from the final.

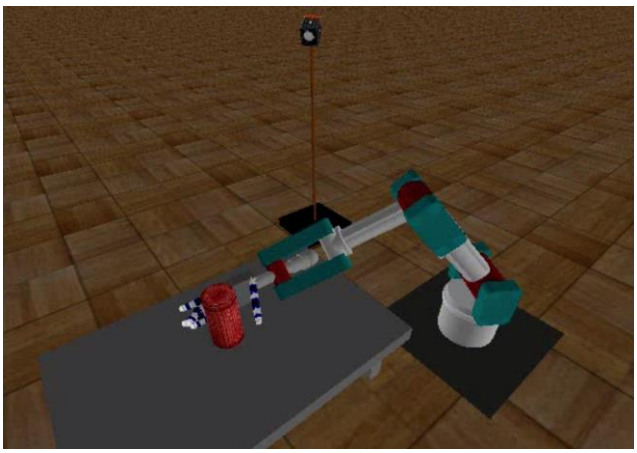

Fig. 3. View of the physics simulator, Open Dynamics Engine (ODE) is used to develop the simulator. The simulated range camera sends a 176x144 ray array from its center with 0.23-degree angular intervals. The first contact of each ray with any surface is determined using ODE functions. The range value is calculated as the distance between the contact point and ray origin point. Camera noise is simulated as a Gaussian noise with zero mean and 0.2 variance.

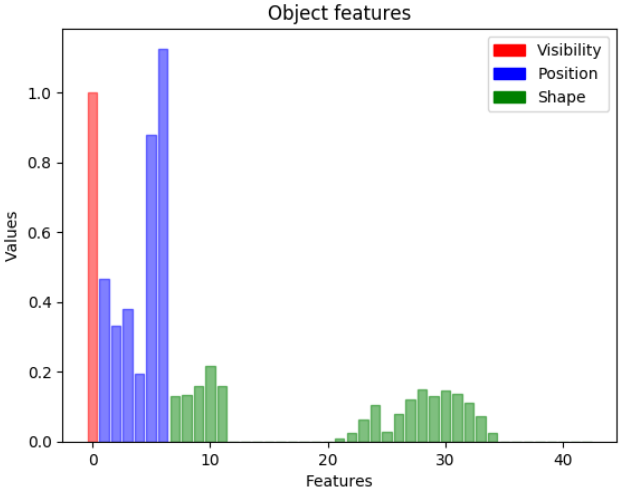

Fig. 4. Bar plot of features of a cylindrical object. Visibility, position and shape-related features are shown with different colors.

4.2. 4.2 Discretization of data

Our aim is to cluster the initial and effect data so that we can form a Bayesian Network. That's why our model takes the initial data and puts it into an autoencoder to reduce dimensionality since clustering algorithms do not perform well in high dimensional data. We choose the K-means algorithm to cluster latent vectors

and 7 is chosen for K because there are 7 types of objects. The result can be seen in Fig. 5.

In the figure, it can be realized that clusters are not separated ideally because of the limited generalization capabilities of the autoencoders. In the ideal scenario, the representation of the object features could be marked by hand. However, the purpose is to create a general framework, which can learn affordances of any random object. Therefore, the objects are not labeled in our input data, we decrease dimensions by using an autoencoder and we cluster the representation using the K-means algorithm.

The K-means algorithm is also applied to effect vectors, but we do not need to put effect vectors into the autoencoder because we figured out later that values of shape vector for effect are not correlated with action type. Thus, it was not necessary to decrease dimensions further. So, omitting shape vectors reduced the dimensionality to 7 from 43. Then the K-means algorithm is applied again and 5 effect categories were obtained.

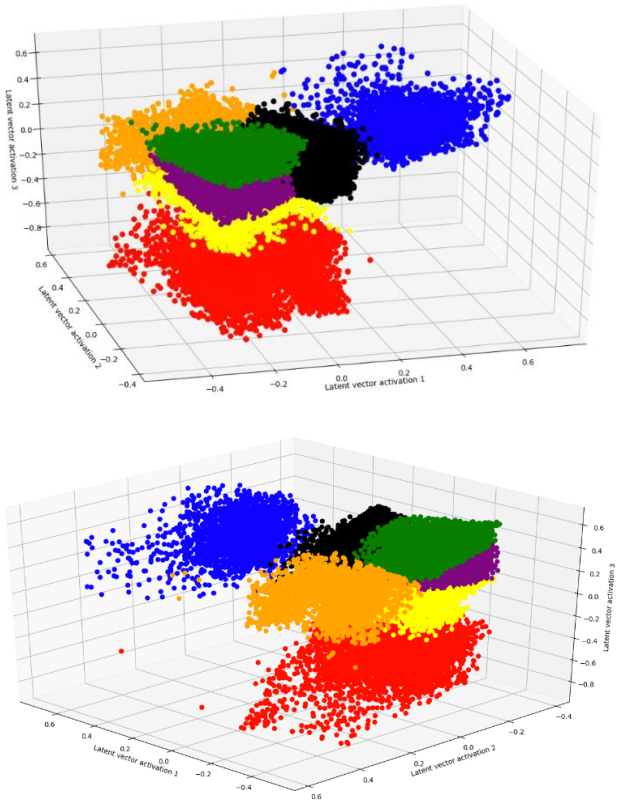

Fig. 5. 7 clusters obtained by the K-means algorithm on object data. The latent space has 3 dimensions. For visualization, the same figure is shown from different view angles. Different colors show different clusters.

4.3. 4.3. Prediction results

After the discretization of the initial and effect vectors, the Bayesian Network is formed. With the Bayesian Network model, it was possible to infer the probability of any node given the other node values. For comparison, three separate SVM classifiers were trained to predict the effect category given the object features and the action type; the object category given the effect features and the action type; and finally, the action type given the object and the effect features.

Table 1: The results for effect, object and action prediction

	Effect prediction	**Object prediction**	**Action prediction**
SVM [29]	83.1%	35.7%	58.6%
Our work	70.3%	48.4%	59%

Note that, originally in [29], for each action one SVM model was trained only to predict the effect. To compare with our model, actions were enumerated as 1000, 0100, 0010, 0001 so that they have equal distance to each other. Then all data for different actions were concatenated and an SVM was trained for effect prediction. Two different SVM models were also required to be trained in the same way for object and action prediction. Furthermore, in [29], action selection was performed by a state space search tree algorithm that predicts effects of different action sequences and selects the closest one.

Bayesian Network provides probabilities for each possible output whereas SVM assigns a single class for a given input. It is both possible to select the highest probability output as prediction (Table 1) or to make a search in outputs with probabilities bigger than a threshold (Table 2).

The results summarized in Table 1 shows that our autoencoder based Bayesian Network model gives similar results in predicting actions, outperforms SVM predictors in predicting objects, and performs worse in predicting effects when the output with highest probability is selected as the prediction of BN. As expected, SVM performs better in predicting the effect as Bayesian Networks lose information while discretizing the initial data. The accuracy fall in object prediction can be interpreted as follows: the prediction becomes an ill-posed problem when SVM tries to predict the initial cluster from initial and effect data. There are potentially different object types that can result in the same effect and it is hard to differentiate them, but Bayesian Networks can model such relations better. The reason for the similarity of action prediction accuracy may be due to the limitations inherent to the experiment setup for action prediction. In the previous work of Ugur et. al [29], it was indicated that all push-actions produced similar effect categories. For example, all 3 pushing actions for spherical objects result in a similar effect which is falling from the table.

Table 2: The results for effect, object and action prediction for different threshold probabilities

	Effect prediction	**Object prediction**	**Action prediction**
SVM [29]	83.1%	38%	58.6%
p(x)>0.33	70.8%	49.1%	59.7%
p(x)>0.25	81.4%	69.2%	84.2%
p(x)>0.15	92.3%	81.3%	93%
p(x)>0	100%	100%	100%

The results summarized in Table 2 shows that our autoencoder based Bayesian Network model gives better results as the threshold probability decreases. Here using the threshold value refers to trying all outputs whose probabilities are larger than the threshold. BN model outperforms the SVM predictor as the threshold probability gets smaller. This approach would decrease the number of states in the search tree of [29] exponentially as well. In [29],

planning was done by constructing a tree structure where nodes and edges represented the perceptual states and action-object pairs, respectively. The sequence of actions and the objects were searched and selected in order to obtain the desired effect. The branching factor in such a search tree is the number of children of each node, and the branching factor was the number of actions × the number of objects. However, the branching factor of the proposed method is proportional to the number of most possible actions and objects for the desired effect. Table 3 illustrates this idea: For a test set with 4288 samples, the method in [29] had to check 4288× 4 actions × 7 objects, in total 120064 leaves but the branching factor of the proposed tree is 7758 for the threshold probability of 0.25. On top of that, the prediction accuracy for the test set is 0.86. In the previous method [29], the accuracy of the planning phase was bounded by the accuracy of the SVM. Here, it is possible to increase the accuracy by decreasing the threshold probability as it is shown in Table 2. In our method, the planning accuracy is bounded by the representation capability of the acquired data about the environment.

Table 3: Results for action and object prediction as a tuple with given effect

	Number of test samples	Number of the searched tuples (branching factor)	Accuracy
p(x)>0.33	4288	4156	0.62
p(x)>0.25	4288	7758	0.85
p(x)>0.18	4288	9460	0.92

One important final remark is that Bayesian Networks provide one unified framework whereas the SVM approach requires a separate model for each variable to predict. Assuming other components such as tools, gestures, language, we would require training and predicting with separate SVMs; however, one single Bayesian Network model with more nodes can be directly used instead.

4.4. Results for effect prediction with missing information

Table 4: Results for effect prediction only with action and only with object

	Effect prediction only with action	Effect prediction only with object
SVM [29]	61.9%	54.8%
Our work	58.1%	53.6%

In the real world, the robots may be in a partially observable state. For example, the camera of the robots may not detect the object properties fully. Thus, we find it important to investigate the robustness of our model in such conditions. We tested the SVM and our model by giving either the object category or the action type as input and requesting the effect category as the output. For these cases, SVM performed slightly better than BN method when output of the BN was used through winner-take-all as it can be seen in Table 4.

On the other hand, BNs inherently provide probabilities along with the predicted values. Therefore, we investigated to exploit provided probabilities and performed a further analysis.

Table 5: Results for effect prediction only with action and only with object using threshold

	Effect prediction only with action	Effect prediction only with object
SVM [29]	61.9%	54.8%
p(x)>0.3	54.1%	25.6%
p(x)>0.25	75%	65.5%

From Table 5, it can be deduced that the prediction of BN significantly improves when the model considers probabilities assigned to outputs and empirically sets a threshold. The results show that using the most probable outcomes that only BN can provide enables our model to outperform SVM. The jump in the part effect prediction only with the object shows that there are many outputs' probabilities within that 0.05 range.

Finally, we should remind again that the old SVM, that was trained before, could not produce these results. For these 2 predictions, 2 more SVMs were trained whereas a single Bayesian Network can make all 5 predictions above.

5. Conclusion

This paper addresses learning affordances and develops a novel method that uses autoencoders to decrease input dimensions in data and constructs a Bayesian Network to encode relationships between action, object and effect categories.

Autoencoders can extract nonlinear and complex relationships and can decrease the number of data dimensions so that the Bayesian Network can be constructed with a few nodes. Since larger BNs have a considerable amount of possible network structures, our network can infer probabilities computationally more efficiently.

In this work, our results are compared with that of Support Vector Machines. The biggest advantage of our approach is that a single Bayesian Network can predict effects, objects, and actions separately but different SVMs are needed to predict different types of outputs. When there is change in input or prediction is done with missing information, new SVMs are required to be trained. However, the Bayesian Network can handle such imperfections. In terms of accuracy, SVM can make equally good or better predictions except for object prediction whereas our model makes its predictions based on the most probable output. However, our model outperforms SVM especially in object and action prediction when it tries all outputs whose probabilities are higher than a threshold. This approach increases the computation time but high accuracy in object and action prediction becomes very useful in the planning phase of a task. When that prediction is not accurate, the model has to consider all possible scenarios to reach the desired effect. In our case, the model can just give the required object and action with high precision. For future work, we will show that this planning approach works well with real robots. We will also investigate to use this approach in more complex object-action interactions.

References

[1] [1] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," Science, vol. 362, no. 6419, pp. 1140–1144, 2018.

[2] [2] G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, "Challenges

of real-world reinforcement learning," arXiv preprint arXiv:1904.12901, 2019.

[3] [3] E. W. Bushnell and J. P. Boudreau, "Motor development and the mind: The potential role of motor abilities as a determinant of aspects of perceptual development," Child Development, vol. 64, no. 4, pp. 1005– 1021, 1993.

[4] [4] K. S. Bourgeois, A. W. Khawar, S. A. Neal, and J. J. Lockman, "Infant manual exploration of objects, surfaces, and their interrelations", Infancy, vol. 8, no. 3, pp. 233–252, 2005.

[5] [5] H. L. Pick, "Eleanor j. gibson: Learning to perceive and perceiving to learn." Developmental Psychology, vol. 28, no. 5, p. 787, 1992.

[6] [6] J. J. Gibson, The Ecological Approach to Visual Perception. Lawrence Erlbaum Associates, 1986.

[7] [7] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action,"Trends Neurosci, vol. 15, pp. 20–25, 1992.

[8] [8] Ben-Gal I., Bayesian Networks, in Ruggeri F., Faltin F. & Kenett R., Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons (2007).

[9] [9] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In Parallel Distributed Processing. Vol 1: Foundations. MIT Press, Cambridge, MA, 1986.

[10] [10] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, E. Thelen, Autonomous Mental Development by Robots and Animals, Science 291 (2001) 599–600.

[11] [11] M. Lungarella, G. Metta, R. Pfeifer, G. Sandini, Developmental robotics: a survey, Connection Science 15 (2003) 151–190.

[12] [12] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, C. Yoshida, Cognitive developmental robotics: a survey, IEEE Transactions on Autonomous Mental Development 1 (2009) 12–34.

[13] [13] A. Stoytchev, Some basic principles of developmental robotics, IEEE, Transactions on Autonomous Mental Development 1 (2009) 122–130.

[14] [14] G. Metta, P. Fitzpatrick, Better vision through manipulation, Adaptive Behavior 11 (2003) 109–128.

[15] [15] P. Fitzpatrick, G. Metta, L. Natale, A. Rao, G. Sandini, Learning about objects through action -initial steps towards artificial cognition, in: Proc. of ICRA 03, IEEE, 2003, pp. 3140–3145.

[16] [16] A. Stoytchev, Behavior-grounded representation of tool affordances, in: In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), IEEE, Barcelona, Spain, 2005, pp. 18–22.

[17] [17] E. Ugur, E. Şahin, Traversability: A case study for learning and perceiving affordances in robots, Adaptive Behavior 18 (2010).

[18] [18] E. Erdemir, C. B. Frankel, K. Kawamura, S. M. Gordon, S. Thornton, B. Ulutas, Towards a cognitive robot that uses internal rehearsal to learn affordance relations, IEEE/RSJ International Conference on Intelligent Robots and Systems (2008) 2016–2021.

[19] [19] G. Fritz, L. Paletta, M. Kumar, G. Dorffner, R. Breithaupt, R. Erich, Visual learning of affordance based cues, in: S. Nolfi, G. Baldassarre, R. Calabretta, J. Hallam, D. Marocco, J.-A. Meyer, D. Parisi (Eds.), From animals to animats 9: Proceedings of the Ninth International Conference on Simulation of Adaptive Behaviour (SAB), LNAI. Volume 4095., Springer-Verlag, Berlin, Roma, Italy, 2006, pp. 52–64.

[20] [20] J. Sinapov, A. Stoytchev, Detecting the functional similarities between tools using a hierarchical representation of outcomes, in: 7th IEEE International Conference on Development and Learning, IEEE, 2008, pp. 91–96.

[21] [21] S. Griffith, J. Sinapov, M. Miller, A. Stoytchev, Toward interactive learning of object categories by a robot: A case study with container and non-container objects, in: Proc. of the 8th IEEE Intl. Conf. on Development and Learning (ICDL), IEEE, Shanghai, China, 2009, pp. 1–6.

[22] [22] I. Cos-Aguilera, L. Canamero, G. M. Hayes, Using a SOFM to learn object affordances, in: In Proceedings of the 5th Workshop of Physical Agents, Girona, Catalonia, Spain.

[23] [23] Y. Demiris, A. Dearden, From motor babbling to hierarchical learning by imitation: a robot developmental pathway, in: Fifth International Workshop on Epigenetic Robotics, Lund University, 2005, pp. 31–37.

[24] [24] S. Hart, R. Grupen, D. Jensen, A relational representation for procedural task knowledge, in: Proceedings of the National Conference on Artificial Intelligence, AAAI Press, 2005, pp. 1280–1285.

[25] [25] L. Montesano, M. Lopes, A. Bernardino, J. Santos-Victor, Learning object affordances: From sensory–motor maps to imitation, IEEE Transactions on Robotics 24 (2008) 15–26.

[26] [26] R. Petrick, D. Kraft, K. Mourão, N. Pugeault, N. Krüger, M. Steedman, Representation and integration: Combining robot control, high-level planning, and action learning, in: P. Doherty, G. Lakemeyer, A. Pobil (Eds.), Proceedings of the 6th International Cognitive Robotics Workshop, 2008, pp. 32–41.

[27] [27] F. Wörgötter, a. Agostini, N. Krüger, N. Shylo, B. Porr, Cognitive agents a procedural perspective relying on the predictability of Object-Action-Complexes (OACs), Robotics and Autonomous Systems 57 (2009) 420–432.

[28] [28] J. Modayil, B. Kuipers, The Initial Development of Object Knowledge by a Learning Robot., Robotics and Autonomous Systems 56 (2008) 879–890.

[29] [29] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," Robot. Autonom. Syst., vol. 59, no. 7–8, pp. 580–595, 2011.

[30] [30] R. M. Haralick, L. G. Shapiro, Computer and Robot Vision, Volume I, Addison-Wesley, 1992.

[31] [31] Giovanni Saponaro, Lorenzo Jamone, Alexandre Bernardino, and Giampiero Salvi. "Beyond the Self: Using Grounded Affordances to Interpret and Describe Others' Actions". In: IEEE Transactions on Cognitive and Developmental Systems (2019). doi: 10.1109/TCDS.2018.2882140 (cit. on pp. 14, 50)

[32] [32] M. Andries, R. O. Chavez-Garcia, R. Chatila, A. Giusti, and L. M. Gambardella, "Affordance equivalences in robotics: a formalism," Frontiers in Neurorobotics, vol. 12, p. 26, 2018. doi: 10.3389/fnbot.2018.00026

[33] [33] Jain, Raghvendra, and Tetsunari Inamura. "Bayesian learning of tool affordances based on generalization of functional feature to estimate effects of unseen tools." Artificial Life and Robotics 18.1-2 (2013): 95-103.

[34] [34] E. Ugur, J. Piater, Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection, IEEE Transactions on Cognitive and Developmental Systems (TCDS), 9(4), pp. 328-340, 2017.

[35] [35] Seker, M. Yunus, Ahmet E. Tekden, and Emre Ugur. "Deep effect trajectory prediction in robot manipulation." Robotics and Autonomous Systems 119 (2019): 173-184.

[36] [36] Chu, Fu-Jen, Ruinian Xu, and Patricio A. Vela. "Detecting Robotic Affordances on Novel Objects with Regional Attention and Attributes." arXiv preprint arXiv:1909.05770 (2019).

[37] Li, Yikun, Lambert Schomaker, and S. Hamidreza Kasaei. "Learning to Grasp 3D Objects using Deep Residual U-Nets." arXiv preprint arXiv:2002.03892 (2020).

[38] Do, Thanh-Toan, Anh Nguyen, and Ian Reid. "Affordancenet: An end-to-end deep learning approach for object affordance detection." 2018 IEEE international conference on robotics and automation

(ICRA). IEEE, 2018.

[39] M. Imre, E. Oztop, Y. Nagai, E. Ugur, Affordance-Based Altruistic Robotic Architecture for Human-Robot Collaboration, Adaptive Behavior, 27(4), pp. 223-241, 2019.