UC San Diego
UC San Diego Previously Published Works

Title
Deep active object recognition by joint label and action prediction

Permalink
https://escholarship.org/uc/item/5xj471nq

Authors
Malmir, Mohsen
Sikka, Karan
Forster, Deborah
et al.

Publication Date
2017-03-01

DOI
10.1016/j.cviu.2016.10.011

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Computer Vision and Image Understanding 000 (2016) 1–10

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Deep active object recognition by joint label and action prediction

Mohsen Malmir[a,*], Karan Sikka[b], Deborah Forster[c], Ian Fasel[d], Javier R. Movellan[d], Garrison W. Cottrell[a]

[a] Computer Science and Engineering Department, University of California San Diego, 9500 Gilman dr., San Diego CA 92093, USA
[b] Electrical and Computer Engineering Department, University of California San Diego, 9500 Gilman dr., San Diego CA 92093, USA
[c] Qualcomm Inst., University of California San Diego, 9500 Gilman dr., San Diego CA 92093, USA
[d] Emotient.com, 4435 Eastgate Mall, Suite 320, San Diego, CA 92121, USA

ARTICLE INFO

Article history:
Received 16 December 2015
Revised 1 October 2016
Accepted 19 October 2016
Available online xxx

Keywords:
Active object recognition
Deep learning
Q-learning

ABSTRACT

An active object recognition system has the advantage of acting in the environment to capture images that are more suited for training and lead to better performance at test time. In this paper, we utilize deep convolutional neural networks for active object recognition by simultaneously predicting the object label and the next action to be performed on the object with the aim of improving recognition performance. We treat active object recognition as a reinforcement learning problem and derive the cost function to train the network for joint prediction of the object label and the action. A generative model of object similarities based on the Dirichlet distribution is proposed and embedded in the network for encoding the state of the system. The training is carried out by simultaneously minimizing the label and action prediction errors using gradient descent. We empirically show that the proposed network is able to predict both the object label and the actions on GERMS, a dataset for active object recognition. We compare the test label prediction accuracy of the proposed model with Dirichlet and Naive Bayes state encoding. The results of experiments suggest that the proposed model equipped with Dirichlet state encoding is superior in performance, and selects images that lead to better training and higher accuracy of label prediction at test time.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

A robot interacting with its environment can collect large volumes of dynamic sensory input to overcome many challenges presented by static data. A robot manipulating an object with the capability to control its camera orientation, for example, is an example of an active object recognition (AOR) system. In such dynamic interactions, the robot can select the training data for its models of the environment, with the goal of maximizing the accuracy with which it perceives its surroundings. In this paper, we focus on AOR with the goal of developing a model that can be used by a robot to recognize an object held in its hand.

There are a variety of approaches to AOR, the goal of which is to re-position sensors or change the environment so that the new inputs to the system become less ambiguous for label prediction (Aloimonos et al., 1988; Bajcsy, 1988; Denzler et al., 2001). An issue with previous approaches to AOR is that they mostly used small simplistic datasets, which were not reflective of challenges in real-world applications (Malmir et al., 2016). To avoid this problem, we have collected a large dataset for AOR, called GERMS,[1] which contains more than 120K high resolution (1920x1080) RGB images of 136 different plush toys. This paper extends our previous work, Deep Q-learning (Malmir et al., 2016), where an action selection network was trained on top of a pre-trained convolutional neural network. In this paper we extend the model to train the network end-to-end using GERMS images to jointly predict object labels and action values.

This paper makes two contributions: First, we develop a deep active object recognition (DAOR) model to jointly predict the label and the best next action on an input image. A deep convolutional neural network is trained to predict the object label and action-values from an image of the object. We use reinforcement learning to teach the network to predict the action values, and minimize the action value prediction error along with the label prediction cross entropy. The visual features in early stages of this network are learned to minimize both errors. The second contribution of

* Corresponding author.
E-mail addresses: mmalmir@eng.ucsd.edu (M. Malmir), ksikka@eng.ucsd.edu (K. Sikka), forster@ucsd.edu (D. Forster), ian@emotient.com (I. Fasel), javier@emotient.com (J.R. Movellan), gary@eng.ucsd.edu (G.W. Cottrell).

[1] Available at http://rubi.ucsd.edu/GERMS/

http://dx.doi.org/10.1016/j.cviu.2016.10.011
1077-3142/© 2016 Elsevier Inc. All rights reserved.

this work is to embed a generative Dirichlet model of objects similarities for encoding the state of the system. This model integrates information from different images into a vector, based on which actions are calculated to improve object recognition. We embed this model as a layer in the network and derive the learning rule for updating the Dirichlet parameters using gradient descent. We conduct a series of experiments on the GERMS dataset to test (1) if the model can be trained jointly for label and action prediction, and (2) how effective is the proposed Dirichlet state encoding compared to more traditional Naive Bayes approach, and (3) discuss some of the properties of the learned policies.

In the next section, we review some of the previous approaches to AOR and the datasets they used. Next we introduce the GERMS dataset and describe the training and testing data used for the experiments in this paper. After that, we describe the details of the proposed network and Dirichlet state encoding, going into the details of cost function and update rules for different layers of the network. In the results section, we report the properties of the proposed network and compare its performance in different state encoding scenarios. The final section is the concluding remarks.

2. Literature review

Active object recognition methods can be divided into two groups based on how they select actions to improve object recognition. The first group uses heuristic methods to select actions, for example to bring the object to a predefined *standard* view where the recognition performance is expected to be maximized. The second group of methods are motivated by information theory, using information gain to determine the effect of actions on object label prediction uncertainty. The next action is chosen to maximize the reduction in this uncertainty.

An early heuristic AOR system was developed by Wilkes and Tsotsos (1992). They used a heuristic procedure to change the camera's position and orientation to bring the object into a 'standard' view using a robotic-arm-mounted camera. The standard view of objects was defined to be unique among all objects with respect to their low level visual features. In a series of experiments on 8 Origami objects, they qualitatively report promising results for achieving the standard view and retrieving the correct object labels. Heuristic method clearly suffer from generalization problem, as the number of objects increases it is not possible to define standard views for each object manually. A more systematic approach is needed to define the effectiveness of different object views for label prediction.

Among the information theoretic approaches to AOR, Schiele and Crowley's work was pioneering in making an analogy between object recognition and information transmission (Schiele and Crowley, 1998). They try to minimize the conditional entropy $H(O|M)$ between the original object O and image M, which is the object's transformation through measurement. Starting from a random view of an object, their system determines the most-likely object label and moves to the view that has the lowest conditional entropy for that label among the training data. The movement is then verified by measuring the prediction discrepancy between the first and the second views. They used the COIL-100 dataset for their experiments, which consists of 7200 images of 100 toy objects rotated in depth (Nayar et al., 1996). This dataset has been appealing for active object recognition because it provides systematically defined views of objects. Schiele and Crowley achieved almost perfect recognition accuracy on this dataset using their one-step view selection procedure.

Borotschnig et al. formulate the observation planning in terms of maximization of the expected entropy loss over actions (Borotschnig et al., 2000). Larger entropy loss is equivalent to less ambiguity in interpreting the image. A set of distributions are learned for different views of each object, and used to predict the entropy loss for the next view. The novelty of this work is the use of parametric distributions for object views. With an active vision system consisting of a turntable and a moving camera, they report improvements in object recognition over random selection of next views on a small set of objects.

Paletta & Pinz search for the most discriminative views of objects by maximizing the entropy loss between two consecutive views of objects (Paletta and Pinz, 2000). The novelty of their method is the use of reinforcement learning to discover the optimal strategies to explore the objects. Action-values in this work correspond to the decrease in entropy of view sequences of objects. A variant of Q-learning is used to train a neural network to predict the action values given the current view of the image. This work is different from our work in that in our model the visual features are learned simultaneously with the optimal policy, which allows the features to be tuned for object inspection. Paletta & Pinz showed that their model is superior in recognizing COIL100 objects compared to a random exploration strategy.

Calculating the exact value for entropy loss is computationally expensive since it requires marginalization over the observation space, and one might resort to approximations or simpler criterion to measure the uncertainty in prediction. This argument motivated Browatzki et al. to maximize a measure of variance of observations across different objects (Browatzki et al., 2014). They used a particle filter approach to determine the viewing pose of an object held in-hand by an iCub humanoid robot. They show that their method is superior to random action selection on small sets of custom objects.

A common trend in these approaches is the use of small, sometimes custom- designed sets of objects. There are medium sized datasets such as COIL-100, which consists of 7200 images of 100 toy objects rotated in depth (Nayar et al., 1996). We have summarized the properties of datasets used in these studies in Table 1. In this table, meridian denotes the great circles on the surface of view sphere of objects, moving along which camera captures images of objects. We also mention the angular distance that camera traverses on the great circle while capturing images, with 2π denoting a full circle. From this table it is clear that these datasets are not challenging for recognition because of small number of objects, simple background and no occlusion of the objects in images. We collected GERMS, which includes a large number of objects with complex background, occlusion and large number of viewing pose per objects to cover the shortcoming of existing AOR datasets.

Another common trend in the existing literature is the notion of a pre-defined encoding scheme for objects appearance. In these studies, visual features extracted from objects are hand-crafted and fixed during policy learning. However, a more compelling scheme would be to learn the features for object appearance encoding along with the object exploration policy. This way we allow the visual features to be fine-tuned for better object inspection. In this paper, we train a deep convolutional neural network to jointly predict label and action-values given objects images. Deep neural networks have proven to be superior in learning visual features to hand-crafted methods. We utilize a deep network to learn the appearance and object inspection policy at the same time. This reduces the training to a single stage, as opposed to the two stage process of feature encoding and policy learning in the current AOR literature.

3. The GERMS dataset

The GERMS dataset was collected in the context of the RUBI project, whose goal is to develop robots that interact with toddlers in early childhood education environments (Malmir et al., 2013; 2016; Movellan et al., 2014). This dataset consists of 1365 video

Table 1
Details of different object datasets used in the literature.

Dataset	Number of objects	Meridians on view sphere	Occlusion	Publicly available	Complex background
Origami objects (Wilkes and Tsotsos, 1992)	8	1(single view)	No	No	No
COIL100 (Nayar et al., 1996)	100	$1 \times 2\pi$	No	Yes	No
model objects (Borotschnig et al., 2000)	15	$3 \times 2\pi$	No	No	No
office objects (Browatzki et al., 2014)	18	Not specified	Yes	No	No
GERMS (Malmir et al., 2016)	**136**	$\mathbf{10} \times \pi$	**Yes**	**Yes**	**Yes**

Fig. 1. The GERMS dataset. The objects represent human cell types, microbes and disease-related organisms.

Table 2
GERMS dataset statistics (mean± std).

	Number of tracks	Images per track	Total number of images
Day 1	816	157 ± 12	76,722
Day 2	549	145 ± 19	51,561

recordings of give-and-take trials using 136 different objects. The objects are soft toys depicting various human cell types, microbes and disease-related organisms. Fig. 1 shows the entire set of these toys. Each video consists of the robot (RUBI) bringing the grasped object to its center of view, rotating it by 180° and then returning it. The dataset was recorded from RUBI's head-mounted camera at 30 frames per second.

The data for GERMS were collected in two days. On the first day, each object was handed to RUBI in one of 6 pre-determined poses, 3 to each arm, after which RUBI grabbed the object and captured images while rotating it. The robot also captured the positions of its joints for every capture image. On the second day, we asked a set of human subjects to hand the GERM objects to RUBI in poses they considered natural. A total of 12 subjects participated in test data collection, each subject handing between 10 and 17 objects to RUBI. For each object, at least 4 different test poses were captured. The background of the GERMS dataset was provided by a large screen TV displaying video scenes from the classroom in which RUBI operates, including toddlers and adults moving around.

We use half of the data collected in day 1 and 2 for training and the other half of each day for testing. More specifically, three random tracks out of six tracks for each object in Day 1 and two randomly selected tracks for each object from Day 2 were used for training the network and the rest was used for testing. Table 2 shows the statistics of training and testing data for the experiments in this paper.

4. Network architecture

The traditional view of an active object recognition pipeline usually treats the visual recognition and action learning problems separately, with visual features being fixed when learning actions. In this work, we try to solve both problems simultaneously to reduce the training time of an AOR model. By incorporating the errors from action prediction into visual feature extraction, we hope to acquire features that are suited for both label and action prediction.

The network architecture is shown in Fig. 2. The input image is first transformed to a set of beliefs over different object labels by a classification network. The belief vector is then combined with the accumulated belief vectors over previous views to produce an encoding of the *state* of the system. This is accomplished by the *Mixture belief update* layer in the network. The new accumulated belief is then transformed into action-values, based on which the next object view is selected.

We next detail each part of the network, describing the challenges in training the layer and corresponding solutions. We first address the transformation of images into beliefs over object classes. Then we outline the belief accumulation problem over object views, followed by the action learning and, finally, present the full description of the algorithm to train this model.

4.1. Single image classification

The goal of this part of the network is to transform a single image into beliefs over different object labels. The feature extraction stage is comprised of 3 convolution layers followed by 3 fully connected layers. The dimensions of each layer are shown in Fig. 2. The convolution layers use filters of size $3 \times 7 \times 7$, $64 \times 5 \times 5$ and $128 \times 3 \times 3$ respectively for layers 1, 2 and 3. The number of parameters in each layer of the network is shown in Table 3. The operations of each layer are inspired by the model proposed in Krizhevsky et al. (2012). Each convolution layer is followed by rectification, normalization across channels and max pooling over a neighborhood of size 2×2 with stride of 1.

We shall denote the GERMS dataset by $D = \{I_i, y_i, P_i\}_{i=1}^{N}$, where $I_i \in \mathbb{R}^{64\times64\times3}$ is the image captured by the robot camera, $y_i \in \{o_1, o_2, \dots, o_C\}$ is the object label and P_i is a positive integer number denoting the pose of the robot's gripper (Malmir et al., 2016). In order to learn the weights of the single image classification part, we perform gradient decent on action prediction and cross-entropy costs, denoted by $\mathbb{C}_{RL}$ and $\mathbb{C}_{CL}$ respectively. The cross-entropy

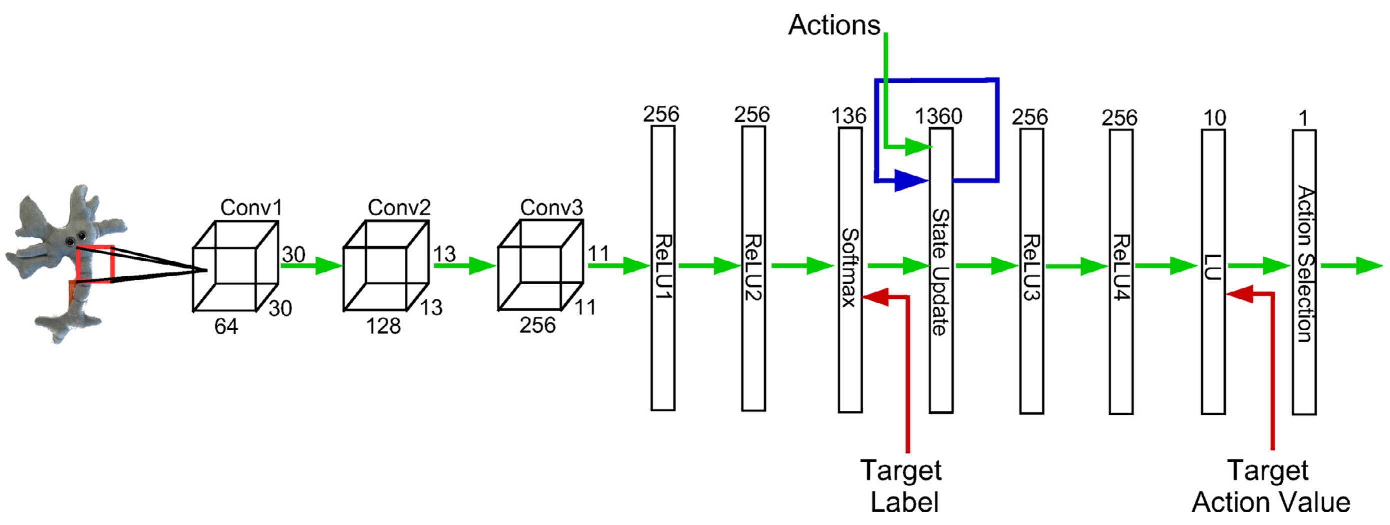

Fig. 2. The network architecture for active object recognition. Red arrows represent target values that are used to train the network. The numbers represent the number of units in each layer of the network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Number of units and parameters for the proposed network.

Layer	Number of units	Input to unit	Num. parameters
Conv1	64 × 30 × 30	3 × 7 × 7	9K
Conv2	4128 × 13 × 13	64 × 5 × 5	204K
Conv3	256 × 11 × 11	128 × 3 × 3	294K
ReLU1	256	30,976	7M
ReLU2	256	256	65K
Softmax	136	256	34K
State Update.	1360	136	184K
ReLU3	256	1360+256	413K
ReLU4	256	256	65K
LU	10	256	2K

classification cost $\mathbb{C}_{CL}$ is:

$$\mathbb{C}_{CL} = -\sum_{i=1}^{N}\sum_{j=1}^{C}\mathbb{I}(y_i = c)\log B_{ij}. \tag{1}$$

Here $\mathbb{I}$ is the indicator function for the class of the object and $B_{ij} = P(o_j|I_i)$ is the predicted label belief for the *i*th image belonging to the *j*th object class. The next subsection describes the action prediction cost $\mathbb{C}_{RL}$.

4.2. Action value prediction

Active object recognition can be treated as a reinforcement learning problem, whose goal is to learn an optimal policy π^*: $S \to A$ from states S to actions A. The optimal policy is expected to maximize the total reward for every *interaction sequence* $s^{\pi}_{0:T}$ with the environment,

$$s_0 \xrightarrow{\pi(s_0)} s_1 \xrightarrow{\pi(s_1)} s_2 \xrightarrow{\pi(s_2)} \ldots \xrightarrow{\pi(s_{T-1})} s_T$$

where $s_i \xrightarrow{\pi(s_i)} s_{i+1}$ is the transition from s_i to s_{i+1} by performing the action $a_i = \pi(s_i)$. The *total reward* for an interaction sequence $s^{\pi}_{0:T}$ is $TR(s^{\pi}_{0:T}) = \sum_{t=0}^{T}\gamma^t R(s_t)$ where $R: S \to \mathbb{R}$ is a reward function and $\gamma, 0 < \gamma < 1$ is a discount factor used to emphasize immediate rewards. For an AOR system, an interaction sequence starts by observing image of the object with the initial orientation in the robot's gripper. The state of the system is then updated by the observed image, and an action is selected to perform on the object to maximize the total reward. The reward in each step is determined by the accuracy of predicted label for the observed images up to that step.

In order to learn the optimal policy, we use the $Q(\lambda)$ algorithm to train the network to predict actions for improved classification (Watkins, 1989). This is a model-free method that learns to predict the expected reward of actions in each state. More specifically, let $Q^{\pi}(s, a)$ be the *action value* for state *s* and action *a*,

$$Q^{\pi}(s,a) = E_{\pi}\{TR(s^{\pi}_{0:T})|s_0 = s, a_0 = a\},$$

which is the expected reward for performing action *a* in state *s* and then following policy π. Let the agent interact with the environment to produce a set of interaction sequences $\{s^{\pi}_{0:T}\}$. Then $Q(\lambda)$ learns a policy by applying the following update rule to every observed transition $TR^{\pi}(s_t, s_{t+1}) = s_t \xrightarrow{\pi(s_t)} s_{t+1}$,

$$Q^{\pi}(s_t,a_t) \leftarrow (1-\alpha)Q^{\pi}(s_t,a_t) + \alpha\left[R(s_{t+1}) + \gamma\max_a Q^{\pi}(s_{t+1},a)\right] \tag{2}$$

where $0 < \alpha < 1$ is the learning rate, and action a_t is selected using an epsilon-greedy version of the learned policy. We interpret this iterative update in the following way to be useful for training a neural network. Let the output layer of the network predict $Q(s, a)$ for the learned policy π for every possible action *a* in *s*. Then a practical approximation of the optimal policy is obtained by minimizing the reinforcement learning cost,

$$\mathbb{C}_{RL} = \sum_{TR^{\pi}(s_t,s_{t+1})\in\{s^{\pi}_{0:T}\}}\left[R(s_{t+1}) + \gamma\max_a Q^{\pi}(s_{t+1},a) - Q^{\pi}(s_t,a_t)\right]^2 \tag{3}$$

In this network, action value prediction is performed by transforming the state of the system s_t at *t*th step through layers ReLU3,ReLU4 and LU. We train the weights of the network in these layers by minimize $\mathbb{C}_{RL}$. In the next subsection, we go into the details of state encoding, and after that we describe the details of the set of actions.

4.3. State encoding

State encoding has a prominent effect on the performance of an AOR system. Based on the current state of the system, an action is selected that is expected to decrease the ambiguity about the object label. An appealing choice is to transform images into beliefs over different target classes and use them as the state of the system. Based on the target label beliefs, the system decides to perform an action to improve its target label prediction. What we expect from the AOR system is to guide the robot to pick object views that are more discriminative among target classes.

We first transform the input image I_i into a belief vector $B_i = [B_{ij}]_{j=1}^{C}$ using the first 7 layers of the network, where

$$B_{ij} \geq 0, \sum_{j=1}^{C} B_{ij} = 1,$$

The produced label belief vector is then combined with the previously observed belief vectors from this interaction sequence to form the state of the system. The motivation for this encoding is that the combined belief encodes the ambiguity of the system about target classes and thus can be used to navigate to more discriminative views of objects. Active object recognition methods usually adapt a Naive Bayes approach to combining beliefs from different observations. Assume that in an interaction sequence, a sequence of images $I_{0:t} = \{I_0, I_1, \ldots, I_t\}$ have been observed and their corresponding beliefs $B_{0:t} = \{B_0, B_1, \ldots, B_t\}$ have been calculated. The state of the system at time t is calculated using Naive Bayes belief combination, which is to take the element-wise product of the individual belief vectors and then normalize,

$$\begin{aligned} s_t = P(O|I_{0:t}) &= \frac{P(O, I_{0:t})}{P(I_{0:t})} \\ &\propto P(O)\prod_{i=0}^{t} P(I_i|O) \\ &\propto \prod_{i=0}^{t} P(O|I_i) \end{aligned} \tag{4}$$

where O is the target label, and $P(O|I_i)$ is the vector of beliefs produced using single image classification. Here we assumed a uniform prior over images and target labels. The problem with Naive Bayes is that if an image is observed repeatedly in $I_{0:\,t}$, the result will change based on the number of repetitions. This is undesirable since the state of the system changes with repeated observations of an image where no new information is added to the system. If a specific image is suitable for classification, the system can visit that image more often to artificially increase the performance of the system. To avoid this problem, we adapt a generative model based on Dirichlet distribution to combine different belief vectors.

We use a generative model similar to Rebguns et al. (2011) to calculate the state of the system given a set of images. The intuition behind this model is that performing an action on an object will produce a distribution of belief vectors. We model the observed belief vectors given the object and action as a Dirichlet distribution, parameters of which are learned from the data. The model is shown in Fig. 3. Here $a \in \{a^1, a^2, \ldots, a^H\}$ is a discrete variable representing the action from the repertoire of actions, $o \in \{o^1, o^2, \ldots, o^C\}$ represents the object label and $\alpha \in \mathbb{R}^C$ is the vector of parameters of the Dirichlet distribution from which the belief vector $B \in \mathbb{R}^C$ over target labels is drawn,

$$\begin{aligned} P(B|\alpha) &= \mathrm{Dir}(B; \alpha) \\ &= \frac{\Gamma(\sum_{j=1}^{C}[\alpha]_j)}{\prod_{j=1}^{C}\Gamma([\alpha]_j)} \prod_{j=1}^{C} [B]_j^{[\alpha]_j - 1} \end{aligned} \tag{5}$$

The state of the system is calculated by computing the posterior probability of object-action beliefs using the model in Fig. 3. Let $P_a^o(a_i, B_i) = P(o, a|a_i, B_i)$ denote the posterior probability of an object-action pair given the performed action and the observed belief vector. Assuming uniform prior over object and α and a deterministic policy for choosing actions,

$$\begin{aligned} &P(o, a|B) \\ &\quad = \frac{\int_\alpha P(o, a, B, \alpha)d\alpha}{P(B)} \\ &\quad \propto \int_\alpha P(o)P(a|o)P(\alpha|o, a)P(B|\alpha)d\alpha \\ &\quad \propto \int_{\alpha_a^o} Dir(B; \alpha_a^o))d\alpha_a^o \end{aligned} \tag{6}$$

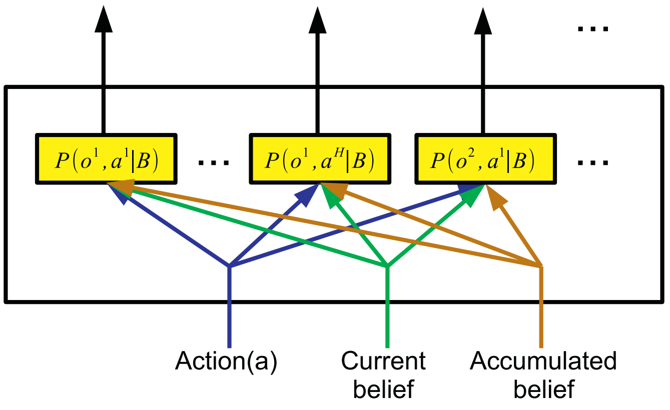

Fig. 3. Dirichlet belief update layer. Each unit in this layer represents a Dirichlet distribution for a pair of object-action. The parameters of this layer are the vectors of Dirichlet parameters α_k^o for each unit.

The notation α_a^o is to make clear that there is an α for each pair of object-action. Instead of full posterior probability, we use $\hat{\alpha}_a^o$, the maximum likelihood estimate of α, and replace the integral above by ,

$$P(o, a|B) \approx Dir(B|\hat{\alpha}_a^o) \tag{7}$$

For an interaction sequence $B_{0:\,t}$ and $A_{0:t} = \{a_0, a_1, \ldots, a_t\}$, the posterior probability of object-action pair is,

$$P(o, a|A_{0:t}, B_{0:t}) = \prod_{i=0}^{t} P(o, a|B_i)^{\mathbb{I}(a, a_i)} \tag{8}$$

The state of the system is comprised of the vector of object posterior beliefs for every object and action, plus the features and belief extracted from the latest image I_t,

$$\begin{aligned} s_t = \{&[P(o, a|A_{0:t}, B_{0:t})], B_t\}, \\ &o \in \{o^1, o^2, \ldots, o^C\} \\ &a \in \{a^1, a^2, \ldots, a^H\} \end{aligned} \tag{9}$$

Note that $s_t \in \mathbb{R}^{CH}$ is a vector of length $C \times H$.

4.4. Training for joint label and action prediction

Our goal is to train the network for joint action and label prediction. We achieve this by minimizing the total cost which is the sum of label (1) and action prediction (3) costs. The errors for action value prediction are back-propagated through the entire network, reaching visual feature extraction units. The total cost function for action-value and label prediction is,

$$Cost = \mathbb{C}_{RL} + \mathbb{C}_{CL} \tag{10}$$

The weights of the network in the visual feature extraction layers (Conv1, Conv2, Conv3, ReLU1, ReLU2, softmax) are trained using backpropagation on (10), while the action prediction layers (ReLU3, ReLU4 and LU) are trained by gradient descent on the action prediction error (3).

We use gradient descent with respect to the network weights to minimize the cost function in (10). If the training converges, it will land on a local optimum point since the neural network's error surface is spiky with many local optimums. A concern may be raised that the minimization of the cost function may diverge, for example if changing the network weights to reduce the reinforcement learning cost causes the classification cost to increase. We didn't observe such behavior in practice while training the network. A counter argument against the divergence of the cost function is that learning a better classifier is in the direction of learning an optimal policy, as less confusion in label prediction simplifies the object exploration policy and can help the policy to more efficiently search for discriminative views of objects.

To learn the parameters of the belief update layer α_a^o, we use gradient descent on log-likelihood of the data. The maximum likelihood of Dirichlet distribution is a convex function of its parameters and can be minimized using gradient descent. For a set of beliefs $B_{1:N}$ observed by performing action a on the object o, the gradient of the log-likelihood with respect to the parameters are,

$$\begin{aligned}\frac{\partial \log P(B_{1:N}|\alpha_a^o)}{\partial[\alpha_a^o]_k} &= N\frac{d}{d[\alpha_a^o]_k}\log\Gamma\left(\sum_{j=1}^{C}[\alpha_a^o]_j\right)\\ &\quad - \frac{d}{d[\alpha_a^o]_k}\log\Gamma([\alpha_a^o]_k) + \log B_k\\ &= N\Psi\left(\sum_{j=1}^{C}[\alpha_a^o]_j\right) - N\Psi([\alpha_a^o]_k) + \log B_k \qquad (11)\end{aligned}$$

where $\Psi(x) = d/d(x)\log\Gamma(x)$ is the digamma function. There is one unit per Dirichlet distribution $Dir(|\alpha_a^O)$ in the belief update layer of the network. These units receive the current belief and the previous state of the system, and produce an updated belief. An schematic of the belief update layer of the network is shown in Fig. 3. Learning α_k^o is carried out simultaneously with the rest of the network weights in the same training session.

4.5. Reward function

Another component that has an important effect on the performance of our AOR system is the reward function which maps state of the system (4) into rewards. A simple choice for reward function is

$$R(s_t) = \begin{cases} +1 & \text{if } \arg\max_i[B_t]_i = \text{Target-Label}(I_t) \\ -1 & \text{otherwise}\end{cases} \qquad (12)$$

A reward of $+1(-1)$ is given to the system if at time step t the action a_t brings the object to a pose for which the predicted label is correct (wrong). The intention behind this reward function is to drive the AOR system to pick actions that lead to best next view of the object in terms of label prediction.

4.6. Action coding

In order to be able to reach every position in the robot's joint gripper range, we use a set of relative rotations as the actions of the system. More specifically, we use 10 actions to rotate the gripper from its current position by any of the following offset values: $\{\pm\frac{\pi}{4}, \pm\frac{\pi}{8}, \pm\frac{\pi}{16}, \pm\frac{\pi}{32}, \pm\frac{\pi}{64}\}$. The total range of rotation for each of the robot's grippers is π. The actions are selected to be fine grained enough so that the robot can reach any position with minimum number of movements possible. This encoding is simple and flexible in the range of positions that the robot can reach, however we found that the policies can become stuck with a few actions without trying the rest. Encoding the states with the Dirichlet belief update helps alleviate this issue to some degree, however, it doesn't completely remove the problem. We deal with this problem by forcing the algorithm to pick the next best action whenever the best action leads to an image which has already been seen.

5. Experimental results

5.1. Training details

We trained the network by minimizing the costs of classification, action value prediction (3) and negative of log-likelihood of Dirichlet distributions (11). We used backpropagation with minibatches of size 128 to train the network. For $Q(\lambda)$, we used initial learning rate of 0.1 which was multiplied by 0.5 after iterations 400,800,1200,1500 and then remained constant. The total number of training iterations is 4000. For each iteration, an interaction sequence of length 5 is followed. The full training procedure is shown in Algorithm 1. For $Q(\lambda)$, we used ϵ-greedy policy in the training stage, with ϵ decreasing step-wise from 0.9 to 0.1. We found that using an $\epsilon > 0$ at the test stage hurts the performance, therefore we used $\epsilon = 0$ during testing. The number of actions is 10 as described above, and there are a total of 136 object classes, resulting in a total of 1360 Dirichlet distributions for state encoding (9).

Algorithm 1 Training the network for joint label and action prediction.

```
1:  procedure TRAIN
2:      $R \leftarrow 1$
3:      for iteration=1 To N do
4:          $I_1, y \leftarrow$ NextImage(iteration)
5:          $s_0 \leftarrow [0]$
6:          Actions ← RandomActions(NumActions)
7:          for t=1 To NumMoves do
8:              $s_t$, predictedActions ← FeedForward($I_t, s_{t-1}$, Actions)
9:              $I_{t+1}, y \leftarrow$ NextImage($I_t$, predictedActions)
10:             targetActionVals, $\hat{y} \leftarrow$ LookAhead($I_{t+1}, s_t$, Actions)
11:             if $t$ = NumMoves then
12:                 targetActionVals ← targetActionVals + $R(s_t)$
13:             for $W \in \{ReLU3, ReLU4, LU\}$ do
14:                 $W \leftarrow W - \lambda_W \frac{\partial}{\partial W}\{\mathbb{C}_{RL}\}$
15:             for $W \in \{Conv1, Conv2, Conv3, ReLU1, ReLU2, Softmax\}$
    do
16:                 $W \leftarrow W - \lambda_W \frac{\partial}{\partial W}\{\mathbb{C}_{RL} + \mathbb{C}_{CL}\}$
17:             for $o \in \{o^1, o^2, \ldots, o^C\}, a \in \{a^1, a^2, \ldots, a^H\}$ do
18:                 $\alpha_a^o \leftarrow \alpha_a^o + \lambda \frac{\partial}{\partial \alpha_a^o} \log P(B_t|\alpha_a^o)$
```

5.2. Learning the parameters of Dirichlet distributions

Fig. 4 shows the average negative log-likelihood of the data under Dirichlet distributions for training a network. This figure shows that the neg-log-likelihood of data decreases for the first 1000 iterations, after which the rate of change is decreased but not stopped. In this figure, there are impulses that occur in the negative-log-likelihood of the data. It is observed that the magnitude of these impulses increase as the model fits the training data. We attribute these impulses to the glitches in the gradients of the action-value cost function with respect to the network weights. As training continues, the glitches are fixed by the image batches for which the network can predict the action-values correctly.

5.3. Label prediction accuracy

5.3.1. Static label prediction

First we report the accuracy of *static label prediction* on GERMS using a deep convolutional network that is trained to predict object labels without the active component. We train a deep convolutional network with 3 convolutional and 2 fully connected layers

Table 4
Comparison of DQN, random and sequential.

Observed frames							
Policy/bserved frame	1	2	3	4	5	6	
Static OR	35.2	46.3	51.0	53.9	56.0	**57.1**	**Right arm**
NB-Rnd	31.3	38.1	41.3	43.4	45.0	46.1	
NB-DAOR	31.3	42.1	45.8	48.0	48.3	49.0	
DR-RND	**40.3**	48.7	51.9	53.6	54.6	55.2	
DR-DAOR	**40.3**	**49.7**	51.6	53.0	52.5	52.6	
DN-RND	39.4	47.8	50.8	52.5	53.6	54.3	
DN-DAOR	39.3	48.4	**53.1**	**55.4**	**57.0**	**57.1**	
Static OR	35.6	46.2	50.8	53.4	55.3	56.6	**Left arm**
NB-Rnd	32.7	39.5	42.9	44.9	46.3	47.4	
NB-DAOR	32.7	43.7	47.5	49.6	50.0	50.6	
DR-RND	43.7	52.5	55.8	57.5	58.6	59.3	
DR-DAOR	43.7	53.0	54.9	55.9	55.5	55.4	
DN-RND	**45.4**	54.5	58.0	60.0	61.1	61.9	
DN-DAOR	**45.4**	**56.3**	**60.7**	**62.8**	**64.1**	**64.6**	

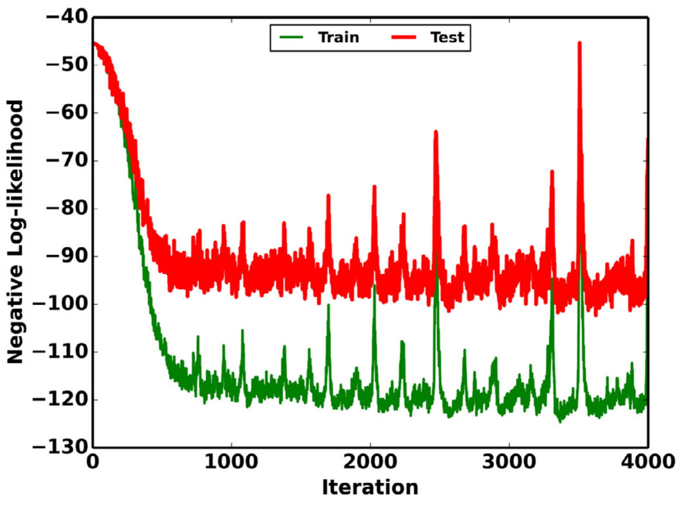

Fig. 4. Average Negative log-likelihood of data under Dirichlet distributions. The decrease in negative log-likelihood indicates learning in the belief update layer.

with the number of units shown in Fig. 2. This network is trained by minimizing the cross entropy cost in (1) for predicting the labels of single frames of the GERMS dataset. Unlike active methods which select different images with different probabilities for training, we select GERMS training images for static label prediction with uniform probability. For testing with one or more images, we randomly select images from each track, and classify them with the trained network. The probabilities for multiple images are combined using the naive Bayes rule in (4). The average accuracy of static label prediction for the test set is shown in Table 4.

We observe that the accuracy of static label prediction is higher than Naive Bayes active methods, but lower than Dirichlet based active models. The difference originates from the ability of active methods in selecting images that are used for training and then to choose such images at test time. Dirichlet based active methods achieve higher accuracy by focusing the training on images that are more discriminative for label prediction. The static model randomly chooses among the ambiguous and non-ambiguous views of different objects at training, which leads to lower accuracy compared to Dirichlet based methods. On the other hand Naive Bayes methods fail to visit enough training images due to overfitting in the action selection layer, and thus are unable to compete in accuracy even with static models.

Since the primary focus of this paper is active object recognition, we do not investigate further the properties of static object recognition models. Instead, we focus on Dicihlet-based and Naive Bayes active object recognition models, and compare their performance in the following sections.

5.3.2. *Comparing Naive Bayes and Dirichlet state encoding*

In this section we compare the effectiveness of the Dirichlet and Naive Bayes state encodings for label prediction accuracy. For Naive Bayes models (NB), the state of the system is updated using (4), while the size and configuration of the rest of the network remain the same. Dirichlet (DR) state encoding is implemented using (9). For each encoding and for each arm, we train 10 different models and report the average test label prediction accuracy as a function of number of observed images, comparing the Deep Active Object Recognition (DAOR) and Random (Rnd) action selection policies. Fig. 5 plots the performance for these models. It is obvious that the Dirichlet model is superior to Naive Bayes in label prediction accuracy.

The first point to notice in Fig. 5 is the performance difference between Naive Bayes and Dirichlet belief updates on single images (action 0). NB models achieve a performance less than 35%, while Dirichlet achieves higher than 40%. One interpretation of this result is that the Naive Bayes model pick actions that bounce between a subset of train images, leading to under-fitting of the model. In the *visualizing policies* subsection, we provide some evidence for this justification. On the other hand, the performance of DR-DAOR model tends to saturate after 3 actions, while DR-Rnd keeps improving for subsequent actions. This might be due to the fact that DR-DAOR also bounces between subsets of images at the test time. We can avoid such behavior by forcing the policies to pick actions that lead to joint poses that haven't already been visited in the same interaction sequence.

5.3.3. *Removing duplicate visits*

We train a set of models using Dirichlet state encoding, while forcing the policy to pick non-duplicate joint poses in every action of an interaction sequence. This approach is easy to implement by keeping a history of visited joint poses during an interaction sequence and picking actions with highest action value that lead to novel joint positions. We refer to this model as Dirichlet with non-repeated visits (DN). Comparison between DN and DR for Rnd and DAOR policies (both forced to visit novel poses) is shown in Fig. 6.

Comparison between the models mentioned above is shown in Table 4. We see that the best performing model is DN-DAOR with the exception of action 1 for the right arm, which DR-DAOR achieves the best performance. For both arms, Dirichlet models perform significantly better than Naive Bayes, improving the model's performance on average by 10% for the right arm and 14% for the left arm.

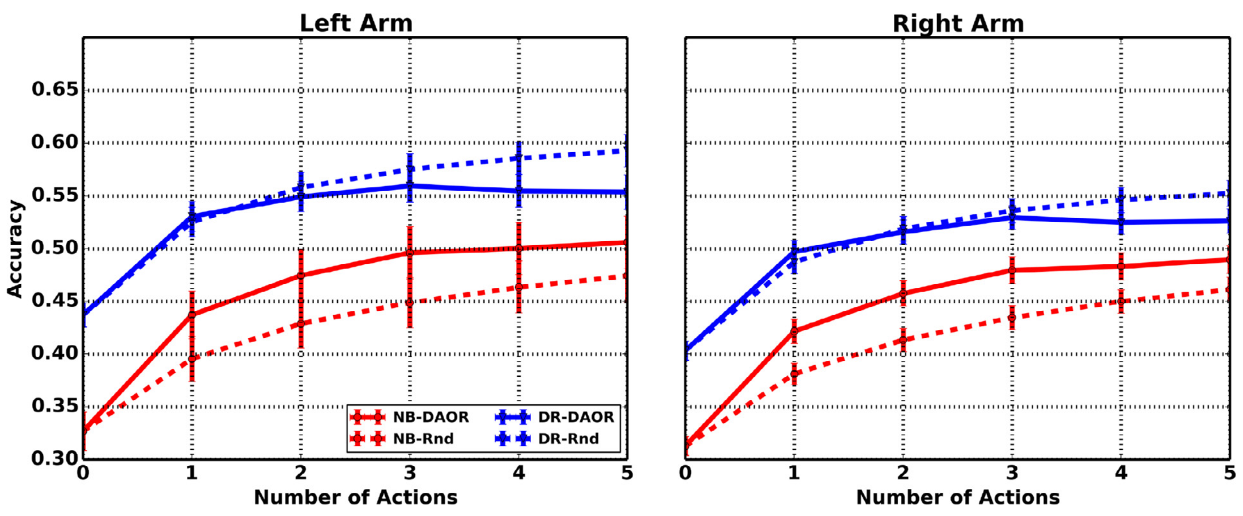

Fig. 5. Test label prediction accuracy as a function of number of observed images for left and right arms for Dirichlet state encoding with repeated visits (DR) and non-repeated visits (DN).

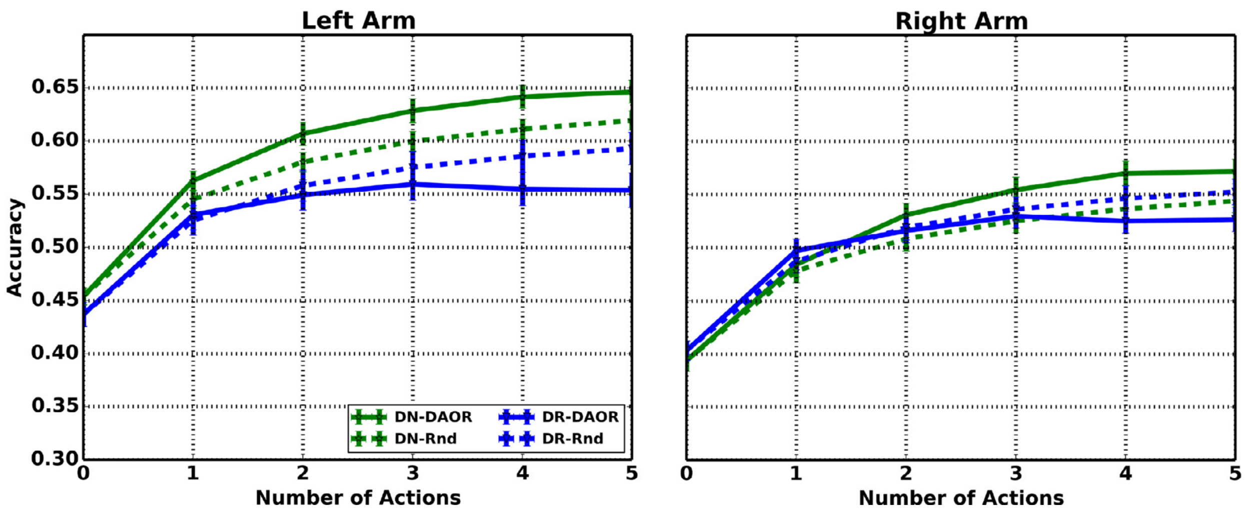

Fig. 6. Test label prediction accuracy as a function of number of observed images for left and right arms for Naive Bayes (NB) and Dirichlet (DR) state encoding.

5.3.4. Visualizing policies

It may help us understand the weakness and strength of different models if we take a closer look into the learned policies. For this purpose, we visualize the consecutive actions in the interaction sequences of length 5, as shown for training data in Fig. 7 and for test data in Fig. 8. Each plot represents actions in different rows, with the magnitude and orientation of the action begin depicted by the length and direction of the corresponding arrow on the left side. Each time step of the interaction sequence is shown as a numbered column. The colored lines in each plot connect one action in column i to another action in column $i+1$ only if those actions appeared consecutively in interaction sequences at these time steps. The thickness of lines depicts the relative frequency by which two actions were observed on the data.

Fig. 7 visualizes the policies DN-DAOR and NB-DAOR on the training data. This figure helps clarify the lower performance of NB models as described before. For NB-DAOR shown on the left side of Fig. 7, we see thick lines connecting actions that rotate the object with the largest magnitude in opposite directions. The relative thickness of these lines indicates that the model tends to go to one end of the joint's rotation range, go back with one large rotation and then repeat. Despite presence of other actions, this back and forth action dominates the training process, leading to lower accuracy on test label prediction for single images. On the right side of Fig. 7 we see that DN-DAOR picks a wide range of actions, which leads to better examination of training images and thus higher performance on single images.

Fig. 8 visualizes the learned policies at test time for NB-DAOR and DN-DAOR. We see on the left side that NB-DAOR only swings between the two large rotations in the opposite direction, while DN-DAOR prefers to do a few larger actions (thick purple and blues lines connecting columns 2, 3 and 4) followed by few smaller actions in different directions. There is no back and forth for DN-DAOR between visited joint positions, which leads to better performance on the test set.

6. Conclusions

In this paper, we proposed a model for active object recognition based on deep convolutional neural networks. The model is trained by minimizing the action and label prediction costs. The visual features in early stages of this network were trained by minimizing both the action and label prediction costs. The difference between the work presented here and deeply supervised networks (Chen-Yu et al., 2014) is that in the latter, the training is carried out by

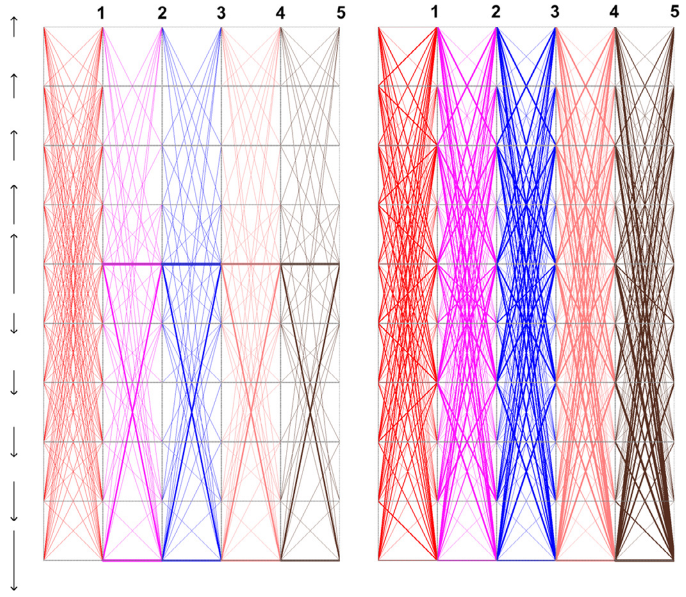

Fig. 7. Visualization of (left) NB and (right) DN policies on training data. Each row represents an action and each column represents a time-step in object exploration performed by the policy in an interaction sequence. The color of lines connecting two columns are different for clarity for every consecutive time steps, while the thickness of these line indicate the frequency of that transition between views in interaction sequences. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

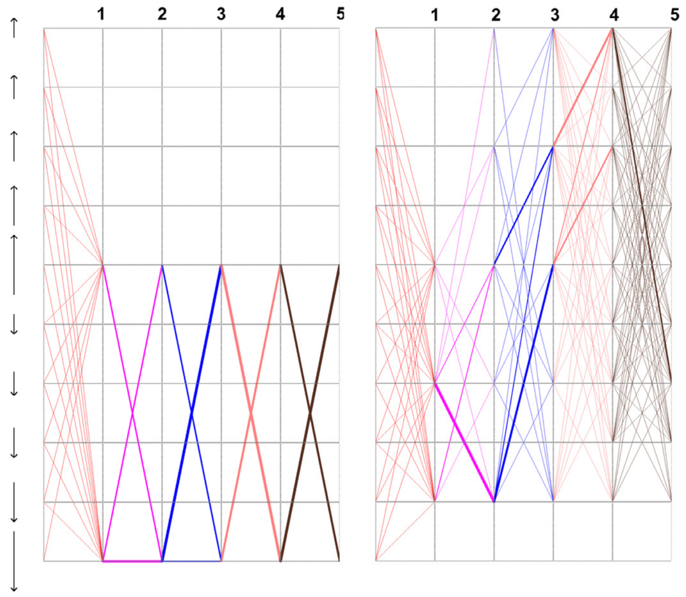

Fig. 8. Visualization of (left) NB and (right) DN policies for test data. NB model prefers to repeats the same two actions, swinging between two joint poses at one end of the joint range. The DN model usually performs a few larger rotations on the object, followed by a few smaller rotations in different directions to inspect the objects in a fine-grained manner.

minimizing the classification error, while in our approach we minimized the action learning cost along with classification error. The joint cost minimization allows the model to learn visual features that are suitable for predicting object label and the action to be performed on the object to improve the recognition performance.

We adapted an alternative approach to the common Naive Bayes belief update rule for state encoding of the system. Naive Bayes has the potential of overfitting to subsets of training images, which could lead to lower accuracy at the test time. We used a generative model based on Dirichlet distribution to model the belief over object-action pairs. This model was embedded into the network, which allowed training the network in one pass jointly with label and action-value learning. The results of experiments confirmed that the proposed Dirichlet model is superior in test label prediction accuracy to the Naive Bayes approach for state encoding.

A common trend we observed in the models trained in this paper was the strong preference for a few actions, which led to limited examination of the objects, and thus lower performance on label prediction. This preference was strongest in the Naive Bayes state encoding models. Employing Dirichlet for state encoding helped alleviate this problem, mainly for the training data and less for the test data. We observed that the strong preference for a limited set of actions weakens for the training stage for the DR-DAOR model, and as a result the model explored the training data more efficiently and achieved higher label prediction accuracy on the test data.

A difficulty that arises in using beliefs for state encoding is the difference in distribution of beliefs over train and test data. This results in overfitting of the policies to high confidence beliefs, which may not be the case for test data. In training our models, the training accuracy reaches above 90% after 1000 iterations. This may cause the algorithm to reward every action, which finally may lead to one action taking over and always producing the highest action value. A remedy for this problem requires the training data to be representative of the test data in prediction accuracy. However we found that the test set in GERMS is very challenging for label prediction. Another possibility is the use of outside data in training the label prediction module, which may help produce more similar distribution of beliefs over training and test data.

Acknowledgments

The research presented here was funded by NSF IIS 0968573 SoCS, IIS INT2-Large 0808767, and NSF SBE-0542013 and in part by US NSF ACI-1541349 and OCI-1246396, the University of California Office of the President, and the California Institute for Telecommunications and Information Technology (Calit2).

References

Aloimonos, J., Weiss, I., Bandyopadhyay, A., 1988. Active vision. Int. J. Comput. Vis. 1 (4), 333–356.
Bajcsy, R., 1988. Active perception. Proc. IEEE 76 (8), 966–1005.
Borotschnig, H., Paletta, L., Prantl, M., Pinz, A., 2000. Appearance-based active object recognition. Image. Vis. Comput. 18 (9), 715–727.
Browatzki, B., Tikhanoff, V., Metta, G., Bulthoff, H.H., Wallraven, C., 2014. Active in--hand object recognition on a humanoid robot. Robotics, IEEE Trans. 30 (5), 1260–1269.
Chen-Yu, L., Saining, X., Patrick, G., Zhengyou, Z., Zhuowen, T., 2014. Deeply-supervised nets. CoRR, abs/1409.5185 3 (4), 93.
Denzler, J., Brown, C.M., Niemann, H., 2001. Optimal Camera Parameter Selection for State Estimation with Applications in Object Recognition. In: Pattern Recognition. Springer, pp. 305–312.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105.
Malmir, M., Forster, D., Youngstrom, K., Morrison, L., Movellan, J., 2013. Home alone: social robots for digital ethnography of toddler behavior. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 762–768.
Malmir, M., Sikka, K., Forster, D., Movellan, J., Cottrell, G.W., 2016. Deep q-learning for active recognition of germs: baseline performance on a standardized dataset for active learning. In: Proceedings of the British Machine Vision Conference (BMVC), pages, p. 161-1.
Movellan, J., Malmir, M., Forester, D., 2014. Hri as a tool to monitor socio-emotional development in early childhood education, in proc. HRI 2nd Workshop on Applications for Emotional Robots, Bielefeld, Germany.
Nayar, S., Nene, S., Murase, H., 1996. Columbia Object Image Library (coil 100). Tech. Rep. Department of Comp. Science, Columbia University, CUCS-006-96.
Paletta, L., Pinz, A., 2000. Active object recognition by view integration and reinforcement learning. Rob. Auton. Syst. 31 (1), 71–86.

ARTICLE IN PRESS

Rebguns, A., Ford, D., Fasel, I.R., 2011. Infomax control for acoustic exploration of objects by a mobile robot. In: Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence.

Schiele, B., Crowley, J.L., 1998. Transinformation for active object recognition. In: Computer Vision, 1998. Sixth International Conference on. IEEE, pp. 249–254.

Watkins, C.J.C.H., 1989. Learning from delayed rewards. University of Cambridge England Ph.D. thesis.

Wilkes, D., Tsotsos, J.K., 1992. Active object recognition. In: Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on. IEEE, pp. 136–141.