This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

Learning Deep Features for Robotic Inference from Physical Interactions

Atabak Dehban, Shanghang Zhang, Nino Cauli, Lorenzo Jamone, and José Santos-Victor

***Abstract*—In order to effectively handle multiple tasks that are not pre-defined, a robotic agent needs to automatically map its high-dimensional sensory inputs into useful features. As a solution, feature learning has empirically shown substantial improvements in obtaining representations that are generalizable to different tasks, compared to feature engineering approaches, but it requires a large amount of data and computational capacity. These challenges are specifically relevant in robotics due to the low signal-to-noise ratios inherent to robotic data, and to the cost typically associated with collecting this type of input.**

In this paper, we propose a deep probabilistic method based on Convolutional Variational Auto-Encoders (CVAEs) to learn visual features suitable for interaction and recognition tasks. We run our experiments on a self-supervised robotic sensorimotor dataset. Our data was acquired with the iCub humanoid and is based on a standard object collection, thus being readily extensible. We evaluated the learned features in terms of usability for 1) object recognition, 2) capturing the statistics of the effects, and 3) planning. In addition, where applicable, we compared the performance of the proposed architecture with other state-of-the-art models. These experiments demonstrate that our model is capable of capturing the functional statistics of action and perception (*i.e.* images) which performs better than existing baselines, without requiring millions of samples or any hand-engineered features.

***Index Terms*—Convolutional Variational Auto-Encoder (CVAE), Representation learning, iCub humanoid robot**

I. Introduction

IEEEPARstartExpressive and concise representations from sensory data are a fundamental requirement of robots that are designed for unstructured environments [1]. These representations should include all the necessary information for the (possibly unknown) robot task while rejecting all the spurious observations. Such tasks can include many computer vision problems *e.g.* object recognition, in addition to more specific learning of sensorimotor mappings.

As an example, one may consider the prediction of the future outcome of actions on the sensory receptors (*i.e.* expected perception) which is an important step for robotic planning and action selection [2]. One way to describe how our actions will influence the environment's behaviour is provided by the laws of physics, and given complete knowledge of the parameters describing such laws, one could predict the motion of objects in everyday situations. However, this detailed physical knowledge is rarely available to agents and obtaining them from robotic interaction data is an active research topic [3], [4]. Indeed, Kaiser *et al.* [5] show that people's beliefs about the motion of objects do not necessarily conform to Newtonian dynamics. *E.g.* many adults fail to realise that if an object is dropped from a moving train, it continues to move in the same direction as the train while falling.

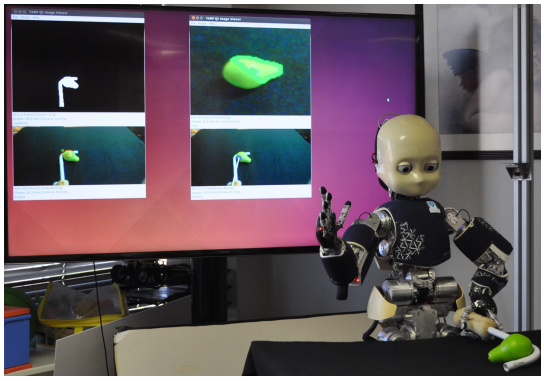


Figure 1: Experimental setup, showing the iCub humanoid robot at the beginning of a robot–object interaction trial, and the reference frame annotation. In the background screen, we show the visual perception routines employed during data collection.

Learning from real-world interactions and observing consequences can offer an alternative route to developing a predictive model of the environment (Fig. 1). Understanding the causal relations between actions and their intermediate perceptual consequences is studied under the concept of *intuitive physics* [6], [7], which explores the common-sense reasoning about simple interactions that happen in the world. In this paper, we propose a deep probabilistic method based on Convolutional Variational Auto-Encoders (CVAEs) to learn generalizable visual representations for multiple robotic tasks, from both interactions (self-supervised) and labels (supervised).

A. Limitations of Existing Work

We categorize the frameworks that attempt to model the behaviour of objects due to external forces into two groups:

This work was supported in part by Fundação para a Ciência e a Tecnologia (FCT) projects under Grant UID/EEA/50009/2020, FCT RBCog-Lab research infrastructure, Lisbon Ellis Unit (LUMLIS), and PhD grant PD/BD/105776/2014. *(Corresponding author: A. Dehban.)*

A. Dehban and J. Santos-Victor are affiliated with the Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisbon, Portugal (e-mail: {adehban,jasv}@isr.tecnico.ulisboa.pt).

S. Zhang is affiliated with the Computer Science Department of Peking University (e-mail: shanghang@pku.edu.cn).

N. Cauli is affiliated with the Department of Mathematics and Computer Science of the University of Catania (e-mail: nino.cauli@unict.it).

L. Jamone is affiliated with the Advanced Robotics at Queen Mary, School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK (e-mail: l.jamone@qmul.ac.uk).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

(1) frameworks that predict action outcomes in the image space (*i.e*. the output is immediately an image), and (2) those that predict action outcomes in a pre-defined *or* learned feature space.

Predicting in image space: Recent advances in deep learning have enabled many models to directly reason and predict at a pixel level. One common approach is to use auto-encoders to learn important features in lower dimensions and use them for predictions [8]. Wahlström *et al.* [9] used auto-encoders to learn a model of an inverted pendulum from simulation. Finn *et al.* [10] introduced a spatial auto-encoder, emphasises the object's location. They have also introduced another heuristic that penalizes the acceleration of the learned features over subsequent frames. This model could be applied in a real robot to perform multiple tasks by training separately for each of them. Jonschkowski and Brock [11] proposed 5 important heuristics relating to robotic tasks that can all augment the reconstruction cost of auto-encoders to learn more useful representations. However, many of these heuristics rely on the temporal coherence of the features of the world.

Apart from using auto-encoders, there exist other approaches that can predict what happens to different parts of the environment if an agent applies different forces to it. In a relatively new line of research, most prominently started by Finn *et al.* [12] several models were proposed that are trained on millions of images to predict image motions in subsequent frames. We compared some of the architectures in this topic from an action perspective [13]. However, predicting the position of each pixel over every frame is a very challenging task and requires huge amounts of data [14].

Predicting in feature space: A family of approaches in this category use the concept of affordances [15] to understand how actions can change the environment [16], [17] (Jamone *et al.* [18] reviewed the concept of affordances from a multi-disciplinary stand-point including robotics). In this regard, we introduced applications of auto-encoders in predicting the future position of an object given the visual features of the object and available tools, together with the action performed by a humanoid robot [19], [20]. Mottaghi *et al.* [21] used images that are accompanied by their 3D pose and simulated interactions in a physics engine. Ahmetoglu *et al.* [22] used an encoder-decoder architecture and showed the emergence of symbols from robot interactions in simulations. The object features were learned from object-centred depth images, though the effects were defined as translational movement and the robot-sensed forces. This is another example of using engineering features in order to maximally benefit from available data and even though the sensing of objects was collected in continuous space, discrete symbols suitable for planning have emerged relating the object's response to executed actions. However, constraining the state-space of effects to translations and forces neglects other aspects such as rotations or topples and thus, not all action-related effects are captured. Another interesting approach is proposed by Agrawal *et al.* [23], in which the authors provide predictions in a feature space that is learned and can guide actions. However, since this learned feature space is not used to reconstruct original images, in order to generalize to a new task (*e.g*. object recognition), they need to incorporate the new considerations and train a new model. Similar to our study, Ha and Schmidhuber [24] also used Variational Auto-Encoders (VAEs) for feature learning in a reinforcement learning scenario to solve video games. Their experiments show the applicability of VAE-learned features for control, however, since their environment was simulated, they had access to an unbounded amount of data and they did not need to manually encourage the features to only learn action relevant information (for example to ignore the background).

To summarize, models that directly output an image present an interesting property: the features that they use are a general and compact representation of the image and they can later be employed for other tasks. But because of the difficulties associated with predicting the entire image, they are mostly either deterministic, require a very large dataset, or only work in simulations and game engines. In these models, it is common to incorporate task–specific heuristics that help to simplify the problem.

On the other hand, the approaches that do not provide predictions in image space can build useful models with less amount of data and by the virtue of probabilistic representations, they are better suited to deal with noisy input (*e.g*. see [25]). Nevertheless, they either have to define a state–space model which requires domain knowledge or, if they provide predictions in a learned feature space, that space must be re-learned when a new task is introduced.

B. Contributions

It is possible to build on top of these two schools of methods, by defining a hybrid approach that has the benefits of both image-based and non-image-based predictions. This model must capture important low-dimensional un-correlated (ideally independent) factors of variations in a dataset, in order to reconstruct the image without any predefined heuristics or hand-engineered features. By ensuring the reconstruction ability of the input, the learned features will be useful for multiple tasks without the need to acquire new ones, because they do not need to be "tuned" to any particular task. A model learned using these features should provide visual predictions that are not necessarily pixel-level accurate, yet they shall capture the essential statistics of the task (forward model). As an example of task generalization permitted by this approach, we also trained a classifier that only looks at the learned features and is able to infer the name of the object. In addition, our model can be used to propose an action that achieves a desired sensory state (inverse model).

More specifically, as the first contribution of this work, we have stepped towards the aforestated goals by training a CVAE on a dataset gathered by a real robot. Relying on probabilistic variational techniques, their interpretations are expected to be more robust to noise by design, since they explicitly represent uncertainty and, thus, they are a better choice for robotics applications.

VAEs can choose the number of their latent variables based on the data and the provided capacity: this way, the burden of selecting a fixed number of features by the designer is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

alleviated, which introduces biases and requires a re-design for every new task. This property is considered problematic for many computer vision tasks where the network falls short of using all its capacity for modelling the data [26]. In contrast, here we are promoting how this property can be applied to obtain a data-driven dimensionality reduction, suitable for robotic tasks.

In order to test our model, we used it on our publicly available dataset collected from an iCub humanoid robot [27]. Deep learning techniques will always benefit from the availability of more data. Deep neural networks are notorious for being data-hungry, however, here, our goal is not to feed in raw image data and output pixel-perfect predictions, as this would require a considerably larger dataset of interactions, possibly infeasible with non-industrial robots. In fact, we rely on background subtraction to avoid wasting the capacity of the VAE in representing the background of images and we see in section IV the impact of this decision in automatic dimensionality reduction. Our last contribution is to show that, through a combination of data augmentation techniques, common engineering practices (*e.g.* background subtraction), and probabilistic modelling it is possible to train very large and deep models on datasets that are obtainable and manageable with real non-industry-grade robots, to tackle a very challenging and important problem as described above.

To summarize, this article provides three contributions:

- Proposing a solution to learn task generalizable visual representations from interactions without using heuristics or engineered features;
- Empirically promoting the benefits of VAEs such as (1) data-driven dimensionality reduction, (2) probabilistic representations, and (3) un-correlated features, in learning visual representations from interactions for robots;
- Training a deep architecture capable of solving various robotic tasks without using simulators and game-engines which often struggle with faithfully simulating real-world physical interactions.

The rest of the paper is structured as follows. In Section II we provide a detailed description of the dataset, we report its properties and some statistics related to it. Then, in Section III we describe the ingredients necessary to build the model of understanding the outcomes of actions and learning probabilistic visual representations. The results of our experiments are described in Section IV. Finally, we conclude the paper with Section V and propose some possible future research directions.

II. Dataset

Data-driven learning methods that do not make many prior assumptions on the underlying distribution of the process tend to be data-hungry. Deep learning methods [28] are an extreme example of this observation and their recently achieved remarkable performance can be largely attributed to the availability of big datasets. However, sharing data in robotics is somewhat more challenging because different research groups use various robots with unequal end effectors. This motivated the collection and use of our dataset which is easily expandable, swaps the robot end-effector with easily fabricated tools, and can be re-created by different research groups. To better facilitate the reproducibility of our results, in this work we provide a more detailed explanation and analysis of this dataset, referred to as Tool and Object aFFordances from Interactions (TOFFI). It includes more than 1000 interactions of the iCub humanoid robot with objects selected from the Yale–CMU–Berkeley (YCB) Object and Model dataset [29]. The authors are unaware of any interactive dataset of this scale from robots that are targeted to developmental learning. TOFFI contains the following information:

1) The images of the object on the table before and after doing the action from both cameras;
2) The foreground of these images;
3) The 3D position of the visual Centre of Mass (CoM) of the object on the table at the beginning and at the end of the action execution;
4) The 2D pixel position of the object's visual CoM in the left camera, associated with the above 3D measurement;
5) Visual features of objects and tools on the table from ten different viewpoints.

Details about the data acquisition protocol and visual feature extraction routines are available in Appendix A [1].

Our dataset is about sensorimotor mappings associated with understanding the effects of actions when performed on objects. It is assumed that the statistics of the effect only depends on the object and tools visual appearances and actions. This assumption holds for many objects that the robot normally interacts with, nevertheless, can be broken in several cases. For example, when an object looks rigid initially but in fact is articulated and shatters upon contact or when an object is so heavy that the robot fails to move it.

In each trial, the robot holds one of the designed tools in its left hand, as in Fig. 1. We have considered three tools: **Rake**, **Stick**, and **Hook**. Four categorical *actions* were selected for the robot to perform: **Push**, **Draw**, **Tap-from-left**, and **Tap-from-right** [2]. Combined with the number of tools, the robot performs 12 actions on objects. By taking into consideration the fact that different objects behave differently when being subject to different actions, we have selected 11 objects from the YCB Object and Model dataset. These objects cover a vast range of shapes and colours. Although we emphasise that in the experiments in this paper, the colour information is removed. Fig. 2 shows the distribution of the effects when the robot performs different actions with different objects. From this figure, for example, it is immediately observable that pulling with a stick is pointless because the object will not move (*i.e.* the distribution of the effect is centred around zero).

III. Methods

In this section, we describe our VAE-based pipeline to solve various multi-modal robotic tasks. We first explain the design

[1] TOFFI is accessible from https://vislab.isr.tecnico.ulisboa.pt/datasets_and_resources/#toolaff.

[2] A video of the robot performing the trials can be viewed at https://youtu.be/pKa6GNeBfjk.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

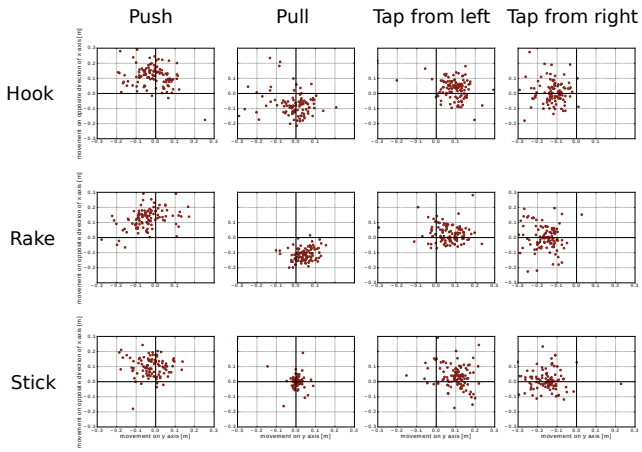


Figure 2: Distribution of the effects measured with the different tools and the actions, as seen from the robot's viewpoint.

of a Convolutional Variational Auto-Encoder (CVAE) to learn important latent features of the robot data. These features are first used in an object recognition task in a supervised manner. Then, we combine the learned features with the action as the *multi-modal* input of a forward-backward model, which is responsible for estimating the changes in the environment caused by the robot's actions. In robotics and machine-learning, multi-modality commonly refers to having more than one sensor modality [30], [31] such as vision, sound, tactile, joint trajectories, *etc*. However, in this paper, we work with high dimensional image data and low-dimensional encodings of the actions. Combining these two sources of information in neural networks is non-trivial as the higher dimensional modality (images) can easily dominate the lower dimensional modality (action encodings) during training. We explain our approach to overcome this difficulty in section III-C.

The model architecture is illustrated in Fig. 3. At a glance, the **Input Image** is a 128×128 grey-scale image of pixel intensities – described in section IV and referred to as $\boldsymbol{x}$ in the rest of this section – that is fed into the CVAE to extract the hidden code, *i.e*. latent variable $\boldsymbol{z}$. This code can be used to reconstruct the image and the convolutional neural networks **Encoder** and **Decoder** are trained according to eq. (2). Afterwards, we select the most informative latent variables in a process explained in section IV-A. These informative latent variables will be used in the bottom part of Fig. 3 together with a one-hot encoded vector representing 1 in 12 **Actions** in the forward and backward models, as explained in section III-C.

A. Variational Auto-Encoders

A Variational Auto-Encoder [32], [33] is a generative model which aims to find latent factors of variation explaining the observations while being as independent as possible. Consider a dataset of observations $\mathcal{D} = \{\boldsymbol{x}^i \mid i = 1, \ldots, N\}$, where each $\boldsymbol{x}^i \in \mathcal{R}^n$. The goal is to find a distribution $q_\phi(\boldsymbol{z}^i \mid \boldsymbol{x}^i)$, $\boldsymbol{z} \in \mathcal{R}^m$ from a family of Gaussian distributions parametrized by mean $\boldsymbol{\mu}_\phi(\boldsymbol{x})$ and standard deviation $\boldsymbol{\sigma}_\phi(\boldsymbol{x})$ that minimizes the Kullback–Leibler divergence (KL divergence): $\mathbb{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^i) \parallel p_\theta(\boldsymbol{z} \mid \boldsymbol{x}^i))$, where $\boldsymbol{z}$ are the latent random variables. Based on the definition of KL divergence, we can derive the following:

$$\log p_\theta(\boldsymbol{x}^i) = \mathbb{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^i) \parallel p_\theta(\boldsymbol{z} \mid \boldsymbol{x}^i)) + \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}^i)}\left[\log p_\theta(\boldsymbol{x}^i \mid \boldsymbol{z})\right] - \mathbb{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^i) \parallel p_\theta(\boldsymbol{z}))}_{\mathcal{O}}$$

Since the KL divergence is positive by definition, we get the lower-bound ($\mathcal{O}$) of the evidence marginal log probability, called Evidence Lower BOund (ELBO):

$$\begin{aligned} \log p_\theta(\boldsymbol{x}^i) \geq & \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}^i)}\left[\log p_\theta(\boldsymbol{x}^i \mid \boldsymbol{z})\right] \\ & - \mathbb{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^i) \parallel p_\theta(\boldsymbol{z})), \end{aligned} \tag{1}$$

In VAEs, $q_\phi(\boldsymbol{z} \mid \boldsymbol{x}^i) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}^i), \boldsymbol{\sigma}_\phi(\boldsymbol{x}^i))$, where $\mathcal{N}$ represents the normal distribution and the parameters $\boldsymbol{\mu}_\phi(\boldsymbol{x}^i)$ and $\boldsymbol{\sigma}_\phi(\boldsymbol{x}^i)$ are approximated with convolutional neural networks which we refer to as the encoder network. The KL divergence term in ELBO (1) tries to minimize the difference between the learned distribution and our prior over the latent variables $p_\theta(\boldsymbol{z})$, which is commonly assumed to follow a standard normal distribution with zero mean and unit variance. This assumption simplifies the computation of the KL divergence, as now it becomes the divergence between standard Gaussian distributions and a Gaussian distribution with a known mean and diagonal covariance matrix, which has a closed-form solution.

The first term in $\mathcal{O}$ measures how well an observation can be reconstructed from the latent variables sampled from the posterior. In the case of Bernoulli observations, the mean of the distribution is approximated via a second dilated convolutional network called the decoder that takes samples from the posterior $q_\phi\left(\boldsymbol{z} \mid \boldsymbol{x}^i\right)$ and outputs the mean value along each dimension of the input (see Fig. 4). Since both the encoder and decoder are implemented as convolutional neural networks, the architecture is called Convolutional Variational Auto-Encoder.

It was shown by Kingma and Welling [33] that if the batch size is large enough, a single sample will be adequate to have a good estimate of the log-likelihood (but see Burda *et al.* [34] for an extension to this). The second term prevents the estimated posterior of the latent variable $\boldsymbol{z}$ from diverging from its prior.

To calculate the gradient of $\mathcal{O}$ with respect to the parameters of the encoder ϕ and decoder θ, the latent variable must be re-parametrized in terms of a base distribution and a differentiable transformation. This can be done via the observation that

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \Longleftrightarrow \boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}),$$

where $\odot$ is element-wise multiplication, $\mathbf{0}$ is a vector of zeros with size m representing the mean and $\boldsymbol{I}$ is a $m \times m$ identity matrix representing the covariance.

The ELBO (1) is an interplay between two terms, where one of them penalizes posteriors that fail to reconstruct the input, and the other one tries to make them as close as possible to an independent standard Gaussian distribution. Although learning factors of variations that are as independent as possible is a desirable property, as noted by Higgins *et al.* [35] it is a strong burden on the learning algorithm and makes the network more difficult to train. To mitigate this problem, a new cost function

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

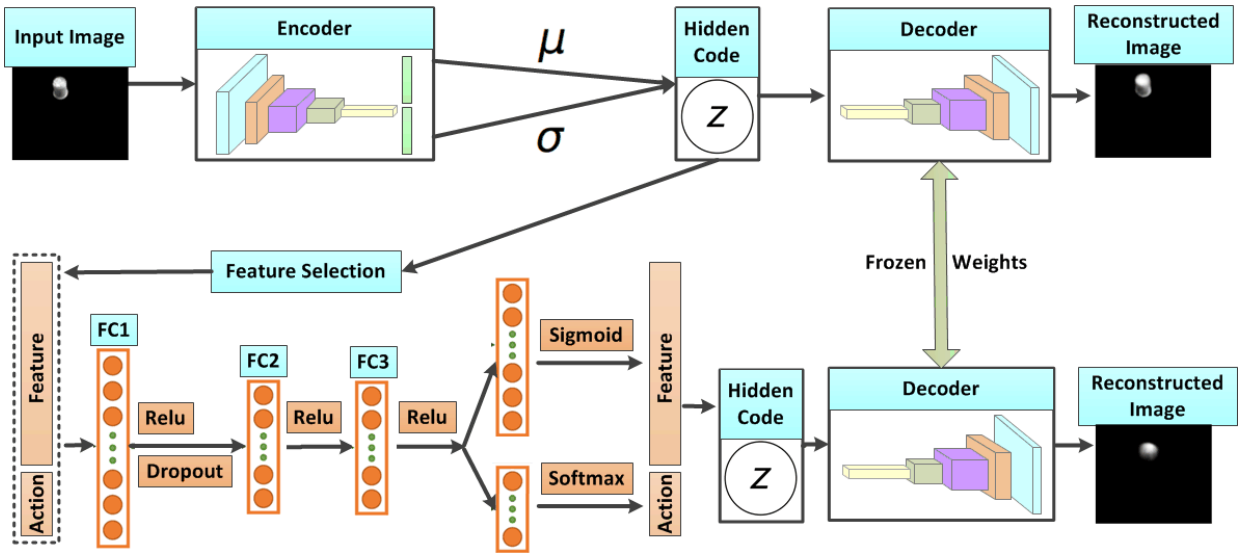


Figure 3: Schematic of the complete model.

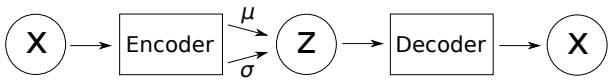


Figure 4: Schematic of the Variational Auto-Encoder. Circles represent random variables, rectangles represent neural networks.

inspired by the negative free energy (notice the sign change) is defined as:

$$\mathbb{L} = \beta \ \mathbb{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p_\theta(\boldsymbol{z})) - \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x} \mid \boldsymbol{z})\right], \tag{2}$$

where β is a hyper-parameter that reflects the desirability of having an independent posterior distribution. If β is zero, then the objective function only tries to maximize the likelihood of the data and by learning to have small values for $\boldsymbol{\sigma}_\phi(\boldsymbol{x})$ the VAE acts almost identical to a conventional auto-encoder. Higgins *et al.* [35] kept β at a constant value and demonstrated the effects that it has on learning powerful latent variables for different datasets. In this work, we have adopted this approach, keeping $\beta = 0.01$ throughout all experiments. Alternatives to keeping $beta$ constant are proposed by Chen *et al.* [36] and Bowman *et al.* [37].

B. Classifier Network

After the Convolutional Variational Auto-Encoder (CVAE) is trained, its weights are frozen for all subsequent experiments and it becomes possible to investigate the latent variables and find important dimensions of variation. The KL divergence term in (2) forces many hidden variables to be generated from the standard Gaussian distribution. The remaining dimensions of the latent variables which have survived the KL divergence part of the loss are later utilized by the decoder network to reconstruct the input. In view of this, we can inspect the variables in the posterior and only select those that have relatively higher KL divergence values across the whole training set to reconstruct the input image. Because the loss function penalizes dependencies between variables, low-dimensional independent factors of variation which are good enough to reconstruct the input start to emerge.

By having an auto-encoder that learns the features of an image, it is possible to use those features in a multitude of tasks, as opposed to scenarios where the features are learned in a task-oriented setting. As the first example of this generalization to new problems, we have used the important latent variables to learn a classifier over objects. We emphasize that the goal of this paper is not object recognition and this network is trained only to provide experimental support for the aforementioned points. It also serves to experimentally validate the benefits of having stochastic visual representations for other downstream tasks.

C. Forward and Backward Models

We have used the same important latent variables of section III-B to train a fully–connected neural network model (called forward model) which predicts how the robot's action affects its future perception and sensory state of the environment. As explained in Section II, we have 4 directional pushes and 3 tools, and we consider each combination as a unique action. By using a one-hot representation, we end up with a vector of size 12 which constitutes the action encoding.

The goal of the forward model is to output an image that is brighter in the probable positions where the object is expected to end up. This is achieved by only predicting the value of important latent variables, and we fill in the rest of the variables by sampling from our prior. Similarly, we train a backward model to retrodict the probable previous appearances of the object, given the current view of it and the applied action. Referring to our second contribution in section I-B, since we are learning probabilistic representations and the important features are stochastic random variables, it is possible to sample from them periodically. This allows an artificial augmentation of datasets with a relatively smaller

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

number of samples, potentially leading to better test time performance.

IV. Implementations and Experimental Results

Here we explain the implementations of the proposed method and extensively evaluate its performance. In order to obtain the input image in Fig. 3, we need the foreground object. To this purpose, we first detect object edges based on the method from Dollár and Zitnick [38]. We learn an accurate edge detector from the inherent structure present in local image patches. A random forest is used to capture the structured information and learned from structured labels to determine the splitting function at each branch in the tree. Each forest predicts a patch of edge pixel labels that are aggregated across the image to compute the final edge map. After the edges are detected, an object bounding box is generated based on the edge information [39], and edges outside the bounding box are filtered out. A threshold is also applied on the edge image to get more accurate object contours. Then, we use a flood-fill operation to fill in the contour and generate the object mask [40]. As a result, the object foreground is obtained based on this mask. A more detailed description of the segmentation pipeline is available in Appendix B.

Our experiments suggest that the more information about the background is available, the more capacity of the network is used to reconstruct it. Even training the same CVAE with the images segmented with the baseline segmentation method of background subtraction and thresholding resulted in an increase of 40% in the number of important latent variables.

The segmented images are then converted to grey-scale and resized to 128×128 resolution. This was motivated by the limitations in the available computing resources. These images are further smoothed out with a Gaussian kernel with a standard deviation of 0.5 with the purpose of mitigating the robot's camera noise. The pixel intensities are scaled between 0 to 1.0 and are modelled as independent Bernoulli distributions given the latent variables $\boldsymbol{z}$ where the mean reflects the intensity of each pixel. These images constitute the dataset $\mathcal{D}$. Because images that are used in this work are very specific and, as such, they are different from natural images, we trained all the networks using the glorot [41] initialization procedure instead of relying on pre-trained models on natural scenes.

A. Reconstruction and Feature Extraction

The top part of Fig. 3 shows the architecture of the CVAE. The details of the network are provided in Appendix C. The CVAE network is trained with the loss function defined in (2).

In order to augment the size of the training set, each image is rotated by 90 degrees three times, which quadruples the size of the training data. We have selected 80% of the images at random to train all the networks, and we have used the remaining 20% for evaluations.

After training, we have calculated the KL divergence term of (2) over our training set. Only 15% of all units have a mean KL divergence that is bigger than the average KL divergence of all units. This sparse representation suggests that many of the hidden units are coming from the prior and the network has automatically selected only enough units to perform a decent reconstruction. Fig. 6 shows the 60 units with the lowest and the 60 units with the highest mean KL divergence.

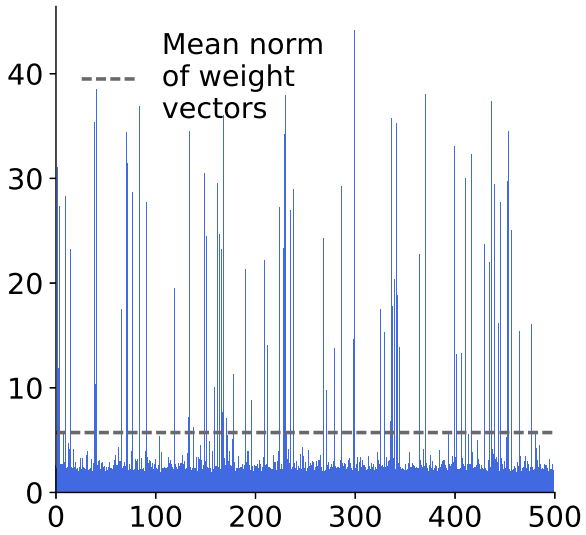


Figure 5: L2 norm of weight vectors associated with each latent variable.

Because the network has a small weight decay of 0.0001, the norm of the weight vector coming out of the latent variables is also sparse (see Fig. 5) and can also be used as an approximate proxy for how informative each hidden variable is. Fig. 7 shows the 60 biggest weight norm vectors along with the weight vectors associated with the highest KL divergence units. The units which are more deviated from the standard Gaussian distribution get bigger weights. This observation is present for almost all 500 hidden units, i.e., the bigger the KL divergence, the bigger is the norm of the associated weight vector.

B. Predicting Object Labels

To gain an understanding of how to use the features learned by the CVAEs, we have trained a deep network to learn object classes from the important components of images. As a baseline, we have only sampled once from important latent variables and trained the network using these samples. By contrast, we have trained another classifier in which at each epoch, *at most* 10% of latent variables were re-sampled and, after 1000 iterations, there will be a new re-sampling for the whole dataset. Both networks are trained for 2000 epochs and all other hyper-parameters are shared between the two networks. The result of this experiment is depicted in Fig. 8.

The network is expected to predict the class of an object from a pool of 11 objects given the sample from important latent variables which were learned from the black and white images by the CVAE in the previous sections. The output of the network goes through a *softmax* layer and the cross-entropy loss is used to train the network. On average, without re-sampling, the network provides an accuracy of 61%, whereas

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

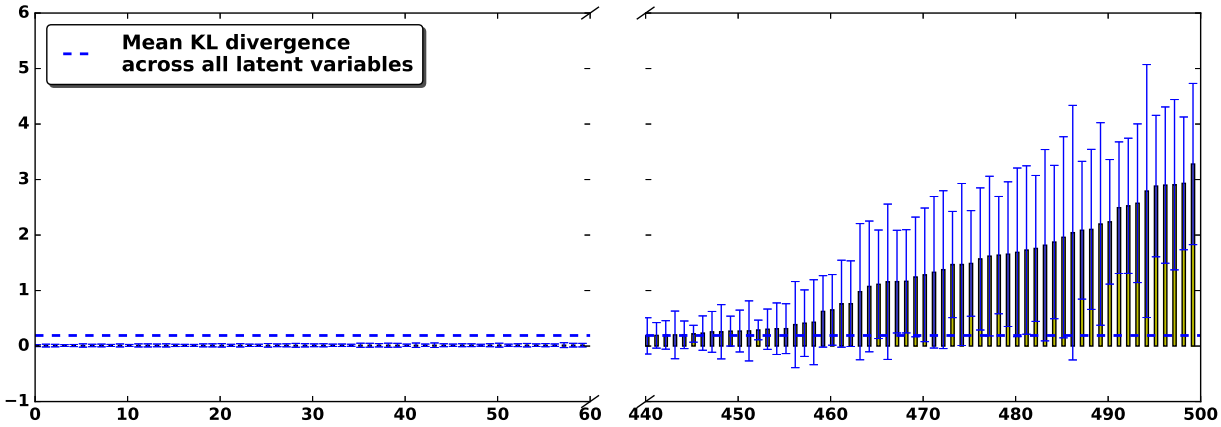


Figure 6: Sorted KL divergence for each latent variable across all samples of the training set.

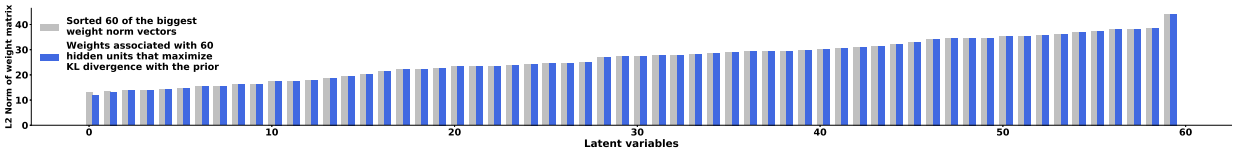


Figure 7: The most informative latent variables, corresponding to the biggest weight vector norm.

by using the introduced re-sampling technique this value can increase to up to 65%. Note that in this setup, we are learning features from down-sampled grey-scale images. This makes it very hard to distinguish between yellow and orange cups, or lemon and the blue ball or the yellow lego, as observed from the confusion matrices. The similarity in sizes, together with the removed colour information makes object recognition very challenging, however, the significantly better than chance performance of the network means that in most cases, the features can be used to distinguish between different classes of objects.

In robotics, it is usually more important for learned features to capture the affordances of objects (roll-able, drag-able, *etc.*) since it is a necessary property for reasoning about how the object would behave in response to an executed action, as evaluated in the next section.

C. Reasoning on the Outcome of Actions

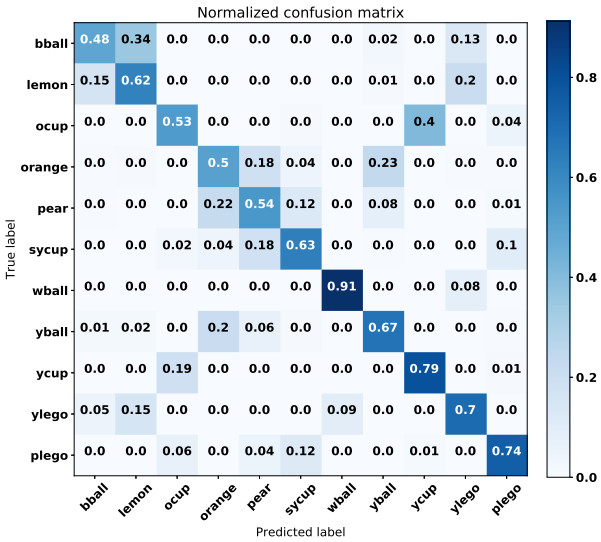
In order to learn how different actions affect the agent's perception of the world, we have selected samples from the previously learned 60 important latent random variables and concatenated them with actions that were performed with different tools. 3 tools and 4 actions were encoded as a one-hot vector of size 12. Together with the samples from the latent variables they make a feature vector of size 72. The target output will also be a vector of size 72, which includes the 60-dimensional sample from important latent variables and the input action; since the action shapes the perception of the robot, we did not want the network to lose sight of this information and force the network to auto-encode the actions as well. Combining multiple input modalities in neural networks is currently challenging (for example, see [42]). One reason is that, just because one modality has bigger dimensions, it encompasses more variation and the network tends to ignore other modalities. When using CNNs, one common approach is to convert every modality to images in order to balance the variations in input modalities [21]. Since the forward and backward models are composed of fully-connected layers, we have constrained the network to auto-encode the actions as well.

The schematic of this architecture is depicted in the lower-left corner of Fig. 3. For more details regarding the structure of the network, we refer the reader to Appendix C. The robot performs more than 1320 trials and, because we are using both cameras, we end up with almost 3000 examples of action executions. We have randomly selected 80% of these trials for training, with the remaining data used for evaluation.

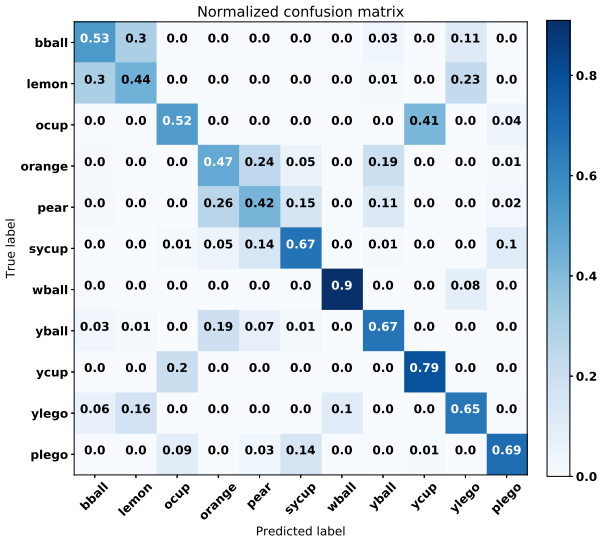
Two models were trained. One *forward* model which sees a sample from the important latent variables of the image (henceforth, features) before applying the action along with the action tag to predict the features of the resulting image together with auto-encoding the action tag. Contrary to this, another *backward* model was trained which sees the features of the image, after the action was applied on it and retrodicts the features of the image before the robot had applied the action. The backward model also auto-encodes the action. The full schematics of these architectures are depicted in Fig. 3. Apart from encoding the actions, the other output of the forward and backward models is the estimation of the 60 important latent variables corresponding to the outcome image with respect to the executed action. The rest of the 440 latent variables are sampled from the prior and the vector of 500 latent variables is fed to the decoder network to generate the outcome image.

In order to obtain a smaller model and to avoid over-fitting, we adopted a *joint* training scheme, in which the two above models share the weights of the aforementioned first two

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems



(a) With re-sampling.

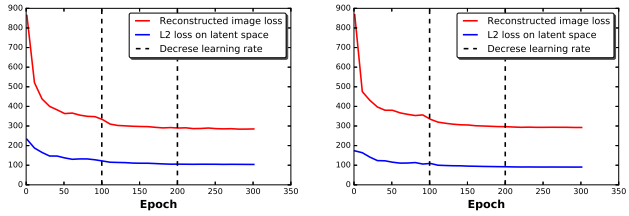


(b) Without re-sampling.

Figure 8: Confusion matrices for object recognition. The numbers are rounded after 2 decimals for readability.

hidden layers, differing only in the last hidden layer and output layer. This way, the majority of the weights become *tied* and are updated with a dataset that has twice the size of the training data. Pinto and Gupta [43] suggested that using one model for learning multiple tasks provides better generalization and the ability to learn better features.

The loss function which was used to train this model is composed of three terms: (1) the cross-entropy loss between the predicted action and the input action; (2) the sum of squared losses between the predicted feature and the target feature; and (3) the cross-entropy loss at pixel level between the predicted image and the target image, after the features have been passed through the decoder network. During training, 50% of each mini-batch contains samples for the forward model, and the other half contains samples for the backward model. Then the loss of each model is calculated



(a) Jointly trained forward model loss;

(b) Jointly trained backward model loss;

Figure 9: Evolution of losses for the forward and backward models.

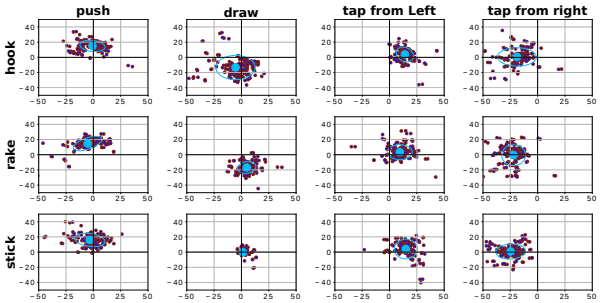


Figure 10: The true movement of the objects in pixel space.

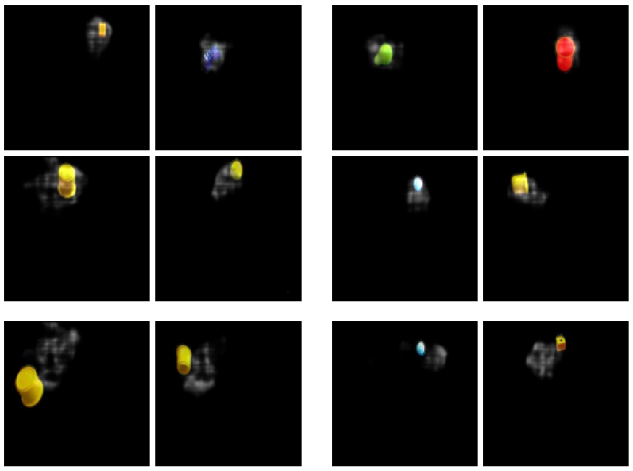
and the variables of forward and backward models are updated according to their associated losses. Fig. 9 shows the two latter losses of the jointly trained forward and backward models. The action loss is not depicted, as it is smaller compared to the other two.

As described in the introduction of this paper in section I, one of the goals of our model is to understand how physical actions affect the perception of the environment. Since the only perceptual modality that our robot currently employs is vision, it should be able to infer the changes caused by its actions in the visual space. Fig. 10 shows how much the visual centre of mass of the object changes by applying different actions. These changes were calculated from images of the train set and, even though they resemble the measured effects in 3D space (see scatter plots of effects in supplementary materials), they are determined in the 2D pixel space.

The actions show trends that are easily observed. For example, in Fig. 10 the action of drawing usually makes objects come toward the robot, however drawing with a hook tool yields a bigger variance on the horizontal axis, whereas when the robot draws objects toward itself with the rake, not only do they come closer, but they tend to go to the right (which is expected, as in our experiments the robot only uses its left arm). The only difference happens when the robot tries to pull objects using a stick, in which case objects move only a little, with a small tendency to shift to the right. Similar observations can be said about other actions of Fig. 10.

A few representative samples from the output result of these two jointly trained models are shown in Fig. 11. In most cases, both networks have learned to spread out a cloud of predictions over the possible locations of the object. It is interesting to note that none of these networks have ever seen any dragged

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems



(a) Forward model predictions. (b) Backward model retrodictions.

Figure 11: Possible object locations (shown as grey-scale points, best seen in colour) computed by the jointly trained forward and backward networks, overlaid with ground truth segmented coloured images. The first two rows are "good" outputs, as the smoky output of the network sufficiently covers the position of the object. The last row shows outputs that do not have enough overlap with the position of the object.

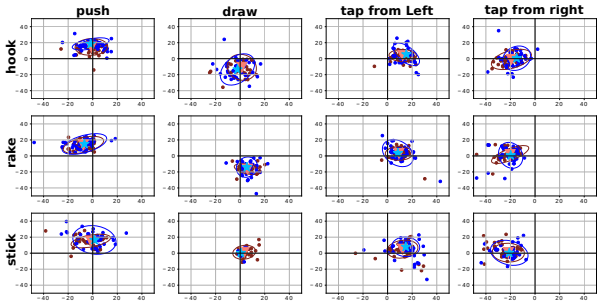
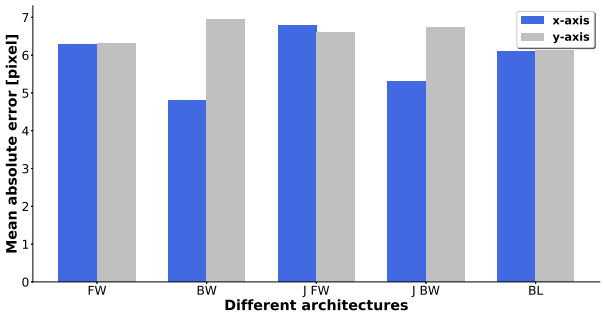


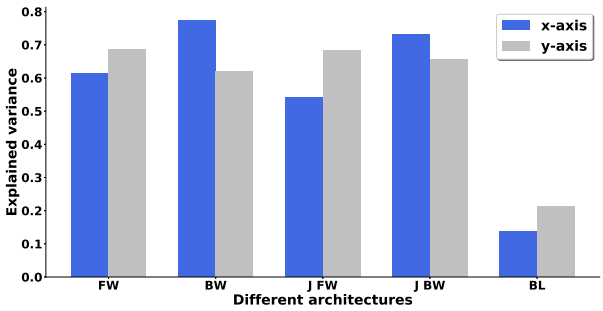
Figure 12: Scatter plots of movement computed by the forward model. Red spots correspond to model outputs, blue spots are the ground truth.

out image, however, they have learned that they can generate these "conservative" predictions/retrodictions by manipulating the 60-dimensional vector of important latent variables. This observation hints that these features can be considered as generative independent factors of variations [44].

Fig. 12 shows the CoM movement output of the jointly trained forward model for test images (the results of the backward model is not presented due to brevity). Because of the smaller sample size divided between each action–tool pair, the *robust* means and covariances [45] are also depicted. Both models have managed to capture the mean and standard deviations reasonably well, making the most identifiable mistake at capturing the correlation coefficient (orientation of the covariance ellipse). It can be explained by the fact that actions are performed along the principal axis and *a priori* we would have expected to observe mainly diagonal covariance matrices. This assumption is better satisfied in Fig. 10, because of the availability of more samples, even though, unlike Fig. 12,



(a) Averaged absolute errors (lower is better).



(b) Explained variances of objects' movements (higher is better).

Figure 13: Evaluation of different architectures averaged across all tool–action pairs. FW: forward model predictions; BW: backward model retrodictions; J: jointly trained; BL: baseline.

Fig. 10 shows *sample* means and covariances.

To have a more quantitative analysis of the performance of the various architectures, Fig. 13 compares the average mean absolute errors made by each architecture. In addition, we also present the result for a baseline, which is using the mean displacement across the train set (visible as the bigger light blue spots in Fig. 10) as the estimated movement.

A thorough inspection of Fig. 13a, which provides the averages of the mean absolute errors across action–object pairs, one may conclude that the baseline does as good a job as any other model by having a comparable mean absolute error. However, this observation can be misleading. The baseline can provide reasonable approximations of displacement, on average, because the first two statistical moments (i.e., mean and covariance) are very similar in the train and test set. Nevertheless, it cannot be a reliable estimator of an object's motion. To show how this baseline fails, for each action–object pair we have sampled from a multivariate Gaussian distribution with the parameters calculated from the train set and depicted in Fig. 10 in light blue, and we used that sample as the effect estimation. Fig. 13b demonstrates how much each model manages to reduce the variance of the observed movement. Even though the baseline still performs better than chance by explaining 10 to 20 per cent of the observed variance (a constant predictor achieves exactly 0% and a bad estimator can even get negative scores), it is significantly overshadowed by other architectures.

Regarding the jointly trained or separately trained schemes that we used in this work, we see no significant decrease in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

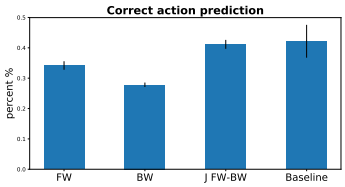


Figure 14: Correct action predictions of different variations of the model. For abbreviations refer to Fig. 13.

performance of the joint model, even though joint training results in smaller models with fewer parameters.

D. Action selection

If a system has truly grasped the statistics of executing actions, the user should be able to exploit that capacity and select actions for achieving a desired effect. In order to quantify how well the model can be used for action selection, we used an experiment similar to [23] in which, given two images, taken before and after executing an action, the network must infer which action was the most probable one that caused the observed effect in the images. This was done by first calculating the movement of the centre of the mass of the object in the two images similar to Fig. 10. Then, given the *before image* and the forward model, we obtain the forward predicted image for all twelve actions and measure the displacement. The action which produces the most similar effect is selected as the target action. A similar procedure can be used for the backward model, or both models can be used simultaneously and the action which on average produces the smallest disparity between the observed and inferred displacement can be selected. The result of this analysis is depicted in Fig. 14.

Each model is trained 100 times and the jointly trained model performs superior to the forward or backward models. The baseline of Agrawal *et al.* [23] was modified to produce a one-hot encoded action and parameters were selected based on the validation set, as the authors have not yet provided the details of their model. As evident by Fig. 14, the jointly trained forward and backward model performs comparably and more reliably than the baseline, even though it was never trained to infer actions. In contrast, the cost function of the baseline directly takes into account the loss in action selection. Additionally, the baseline model is not suitable for task generalization (*e.g.* predicting object labels, section IV-B) without incorporating task-specific information into the training loss of the features.

E. Summary

Reasoning directly on the 2D space of pixels of an image is a difficult task. In order to avoid this problem, we have first trained a Convolutional Variational Auto-Encoder (CVAE) on the images of a dataset. This part of the architecture does not see the actions. It merely gets an augmented set of images and tries to reconstruct them by using samples from the posterior that are penalized to be as independent as possible. We have run further analysis on the learned important latent variables and it turns out that, over the whole training set, only one of those hidden variables has an absolute correlation coefficient bigger than 0.5 with another. *I.e.* the CVAE has learned to decompose the images into disentangled factors of variation.

More importantly, even though we have selected 500 random variables a priori, the network has learned to discard 85% of those variables. These latent features are further used in *three* distinct scenarios. In one task we have used the important features determined by CVAE to classify objects based on their representative latent variables. This task served as an example of the usefulness of the features for several tasks.

In another scenario, we trained a fully-connected neural network to estimate the expected perceptual outcome of an action. Since these latent variables representing images were uncorrelated, it was unnecessary to normalize them before feeding them into the neural network. The result of our experiments with a normalized version of features (unreported) has validated this hypothesis. We have shown that this approach not only captures the statistics of actions but also that the model, on average, makes an error of about 6 pixels in each dimension, which is roughly equal to 5 cm on a table that measures 1 m in latitude and 80 cm in longitude. This result is obtained without defining any explicit state–space model, heuristics or hand-engineered features. Since a fully-connected neural network has a lot of parameters, we have used a multi-task learning approach to train a majority of the parameters of the network with double the size of our train set. These hyper-parameters are reported in the appendix.

Finally, we have used these networks for action prediction. Our experiments show that even without explicitly training the network for inferring actions, it is possible to obtain an accuracy on par with state of the art methods that learn features only for action predictions.

V. Conclusions and Future Directions

Predicting the outcome of actions over the whole perceptual space (put into context, reconstructing the entire expected sensory input) is a challenging task. It is possible to simplify the problem by defining an appropriate state-space or by using some task-related priors. However, a biological agent acting in an environment may not have access to such state-space a priori [46] and must infer it from experience (For a more detailed description, look at windowless–room thought experiment described by [47, p. 258]). We believe that, in order to approach this problem, it is important to define what must be the output of such a system. Arguably, the goal of understanding the outcome of actions would be to guide action selection and, thus, we have defined capturing the statistics of action outcomes in the perceptual space as the desired output of our model. By learning task agnostic features, to the best of our knowledge, a novel formulation of this problem is provided that can learn image features that are useful across multiple tasks.

Given that our perceptual modality is vision, it is not surprising that a deep learning method can provide a good solution to the problem, but perhaps even more surprising is how it was

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

possible to get these results with relatively few data samples. First, we have avoided end-to-end approaches, relying instead on engineering techniques to remove all the non-informative parts of the visual space (namely, the background). Secondly, we have adopted a multi-task learning framework (forward and backward models) where, by learning two disjoint tasks, not only have we increased the training data for a big portion of *model parameters*, but we have also forced them to be useful for different tasks and, as a result, we have reduced the number of trainable parameters.

In the opinion of the authors, the biggest limitation of this work is the need for background segmentation. An end-to-end approach with many thousands of samples appears to be able to learn to reject the background, but that method is not very data-efficient. One way that humans deal with this problem is attention. We think that it is possible to use our segmentation mask to learn an implicit attention model and replace the segmentation pipeline with this model at test time.

Another limitation of the current study lies in the way we represent the actions. By using a one-hot encoded representation, we would need to train new forward and backward models to generalize to novel actions. However, naïvely using joint trajectories is also (in our opinion) akin to using unprocessed images and not useful when the amount of available data is moderate. An engineered representation of actions [48] is a more suitable choice that we will explore in future work.

In this work, the CVAE does not differentiate between true image samples and augmented image samples. In order to use the available data most efficiently, it is possible to utilize only a quarter of hidden variables to reconstruct original images and use the remaining latent variables to reconstruct rotated images, in an approach similar to the one by [49], to further reduce the dimensionality of hidden variables.

Last, in this work, we have adopted a very simple approach for multi-task learning. Our model learns the features for the two tasks at the same time. This could be replaced by the *learning without forgetting* technique introduced by Li and Hoiem [50], which has shown to learn better features for both tasks.

APPENDIX A
TOFFI DATASET DETAILS

To facilitate research in the field of action-conditioned sensory prediction, we have prepared a publicly available dataset, TOFFI which includes more than 1000 interactions of the iCub humanoid robot with objects selected from the YCB Object and Model dataset [29].
This dataset contains the following information:

- Image of the object on the table before and after doing the action from both cameras;
- the foreground of the above images, calculated as explained in Appendix B;
- 3D position of the visual CoM of the object on the table at the beginning and at the end of the action execution;
- 2D pixel position of the object's visual CoM in the left camera, associated with the above 3D measurement;

In Appendix A-A we describe the elements that were used for our data, including a dexterous humanoid robot, everyday tools, and common objects taken from the standard YCB Object and Model dataset. In Appendix A-B we give details about the data acquisition protocol as well as the statistical distribution of the recorded data.

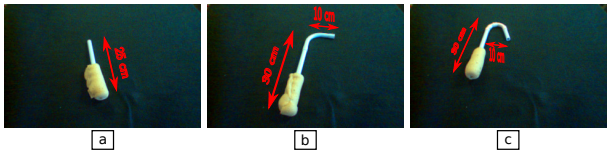


Figure 15: Tools employed for the experiments, with their respective sizes. (a) stick, (b) rake and (c) hook.

A. Robot–Object Interaction Trials

In each trial of our experiments, the robot holds one of the designed tools in its left hand, as in Fig. 1. The transformation from the centre of the palm to the tool's end effector/tip is provided by the experimenter. Gibson defines tools as objects that are detached and rigid [15, p. 40] and we have selected tools that are primitive and built from PVC pipes, in order to be easily re-fabricated. We have considered three tools, shown in Fig. 15:

Rake This tool is most effective when the robot tries to pull objects towards itself or push them away. However, because of the curvature on its left side and its pointy head on the right, the results of tapping left and right with this tool is less predictable.

Stick The stick was selected to be somehow complementary to the rake. Its pointy head makes it less suitable for pushing since small errors in the initial placement of the tool with respect to the object results in the object slipping away. It also cannot be used to draw objects towards the robot. However, it gives the most consistent results when the robot tries to move objects laterally to the left and right.

Hook Trying to add a tool that acts somewhere in between the aforementioned ones, the hook does a good job pulling smaller objects towards the robot but for the bigger objects, it is less reliable. Its asymmetrical shape also results in different outcomes when the objects are pulled to the left or right.

Four categorical *actions* were selected for the robot to perform. Combined with the number of tools, the robot performs 12 actions on objects. The actions are also primitive and consist of drawing the object along the four cardinal directions. In order to make the actions repeatable, the initial position of the tooltip with respect to the object's visual centre of mass is held consistent across all trials for all actions up to the accuracy provided by the robot controllers. Fig. 16 shows how different tooltips are placed with respect to objects before carrying out the actions.

After the tooltip is placed on the table, each action is carried out as the following. The length of movement is kept at 12 cm for all cases:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

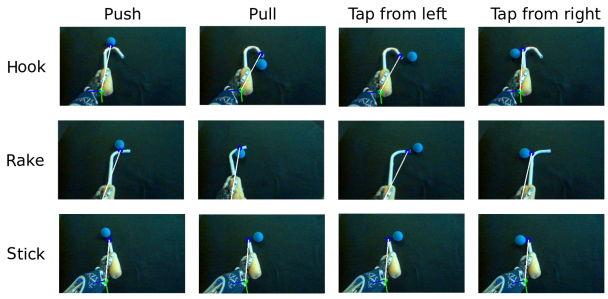


Figure 16: Initial placement of tooltips, in accordance with tool type and action performed by the robot.

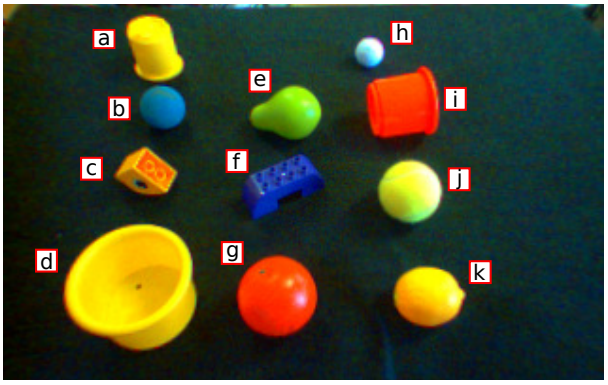


Figure 17: Objects selected for the experiments. Using the abbreviations s–small, y–yellow, b–blue, p–purple, w–white and o–orange, we refer to the objects as: (a) sycup, (b) bball, (c) ylego, (d) ycup, (e) pear, (f) plego, (g) orange, (h) wball, (i) ocup, (j) yball and (k) lemon.

Push	The tooltip is placed below the object and moves away from the robot along the x-axis (see Fig. 1);
Draw	Opposite to pushing, the robot draws the object toward itself and along the x-axis by placing the tooltip on the table over the object and moving it horizontally;
Tap-from-left	For this action, the tooltip is placed on the left side of the object and, by moving it to the right and along the y-axis, the object moves to the right;
Tap-from-right	Unlike the previous action, the robot has to place the tool's end effector to the right side of the object from its own point of view, and drag the object to the left.

By taking into consideration the fact that different objects behave differently when being subject to different actions, we have selected 11 objects from the YCB Object and Model dataset. These objects cover a vast range of shapes and colours and they were selected for two main reasons. First, It was previously shown by Jamone *et al.* [51] that the iCub can manipulate these objects. Second, because they are accessible, other researchers can validate our results or augment the number of trials. In the dataset, each object is identified with a tag, as shown in Fig. 17.

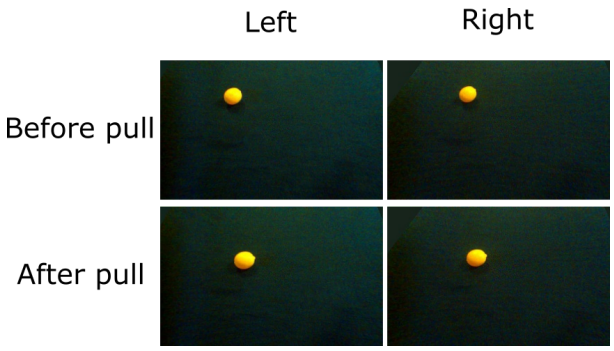


Figure 18: Images resulting from pulling a lemon towards the robot when using the rake tool, as seen from the left and right cameras.

B. Data Acquisition Procedure

Before a trial, the robot takes one image of the object on the table with each camera and records the 3D position of the object as well as its 2D pixel position in the robot's left camera image. Afterwards, the robot proceeds to carry out the action and, upon finishing, it returns to its initial position, it takes one more image with each camera and notes the new position of the object. If the experimenter decides that the action which was executed complies with the definitions in the previous section, then all the images and object positions are saved to disk, otherwise, the buffers will be flushed and the robot prepares for the next trial.

The whole experiment includes information about 3 tools, 4 actions, 11 objects and at least 10 repetitions of each trial, which results in more than 1320 unique trials. Considering that 4 images are recorded during each trial, more than 5280 images are provided. The images were originally taken from the robot with a resolution of 320×240 pixels. The top 40 rows of each image are cropped out, as they corresponded to areas outside of the table, resulting in final images of size 320×200. Further modifications and masking were performed to remove undesirable parts and objects that have appeared during trials. Fig. 18 demonstrates images of an example trial of drawing closer a "lemon object", using the rake tool.

In addition to images, we also extract some visual features of objects and tools from their segmented silhouettes. This procedure and its motivation are explained below.

APPENDIX B
OBJECT SEGMENTATION

Our segmentation procedure was briefly mentioned in section IV, because it is not a major contribution of the paper. However, for completeness, we describe here the segmentation pipeline with more details.

Despite dealing with a single object in a relatively uniform background, segmenting objects accurately from all the images of our dataset, and accurately preserving object contours, is a task that presents several challenges: First, the background presents severe noise and illumination variance (see Fig. 20a); Second, as there is no training data for the foreground segmentation, the deep learning based segmentation methods lose its

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

Table I: Hyper-parameters of all architectures. B: applying batch normalization before the activation function of a layer. [T]C(k, f, s): [Transpose] convolution with kernel size k, number of filters f, and strides s. D(p): Dropout with probability p of keeping activations. FC(n): Fully connected layer with n number of neurons. R(x): Resize x times the original shape with bilinear interpolation.

Network	Hyper-parameters
CVAE Encoder	BC(7,32,2)-BC(5,32,2)-2×BC(5,64,2)-BC(4,128,2)-FC(1000)
CVAE Decoder	BTC(4,128,1)-2×BTC(5,64,2)-BTC(5,32,2)-BTC(5,16,2)-C(5,8,1)-R(2)-C(8,1,1)
Classifier	FC(80)-D(0.7)-FC(30)-D(0.7)-FC(11)
Forward/Backward	FC(80)-D(0.7)-FC(50)-FC(72)

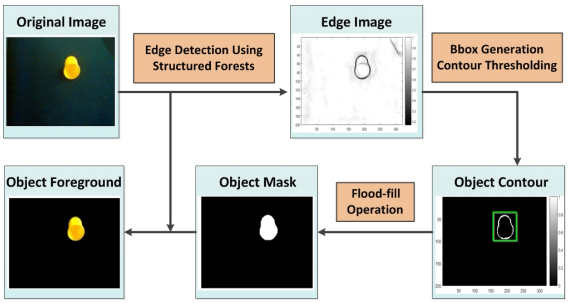


Figure 19: Methodology to segment objects with accurate contours, described in Appendix B.

efficacy on our data; furthermore, the object shadows induce large segmentation errors; In addition, our proposed framework requires that the segmentation process outputs reasonably accurate object contours, yet traditional segmentation methods, often relying on morphology operations such as dilation and erosion, smooth the object contour and reduce the contour segmentation accuracy (see Figs. 20b and 20e).

To overcome these challenges, we design an approach to segment objects with accurate contours. As illustrated in Fig. 19, we first detect object edges based on the method by Dollár and Zitnick [38]. We learn an accurate edge detector from the inherent structure present in local image patches. A random forest is used to capture the structured information and learn from structured labels to determine the splitting function at each branch in the tree. Each forest predicts a patch of edge pixel labels that are aggregated across the image to compute the final edge map. After the edges are detected, an object bounding box is generated based on the edge information, as in [39], and edges outside the bounding box are filtered out. A threshold is also applied on the edge image to get more accurate object contours. Then, we use a flood-fill operation [40] to fill in the contour and generate the object mask. As a result, the object foreground is obtained based on this mask. Fig. 19 depicts the steps used to acquire an object's mask. Some examples of the output of this method, together with simpler baseline segmentation techniques, are shown in Fig. 20.

APPENDIX C
DETAILS OF THE ARCHITECTURES AND TRAINING

In this section, the details of the architectures we used in this work are described. All the hyper-parameters are listed in table I.

A. Convolutional Variational Auto-Encoder

The architecture of CVAE is the upper branch of the whole mode described in Fig. 3. In the CVAE, we have used the ELU activation function [52], except for the outputs of the encoder and decoder. The encoder's output layer for mean values goes through a linear activation function, and for the standard deviation values, we have used the *softplus* function, to guarantee positive standard deviations. The output layer of the decoder has a *sigmoid* activation function. The CVAE is trained for 500 iterations using the ADAM optimizer [53] and mini-batches of size 200. The training begins with a learning rate of 0.01, after 100 iterations it is reduced to 0.001 and after 300 iterations we change it to 0.0001. Starting with a larger learning rate can sometimes help to avoid local minima. However, by continuously reducing the learning rate, the error reaches a stable value.

B. Classifier Network

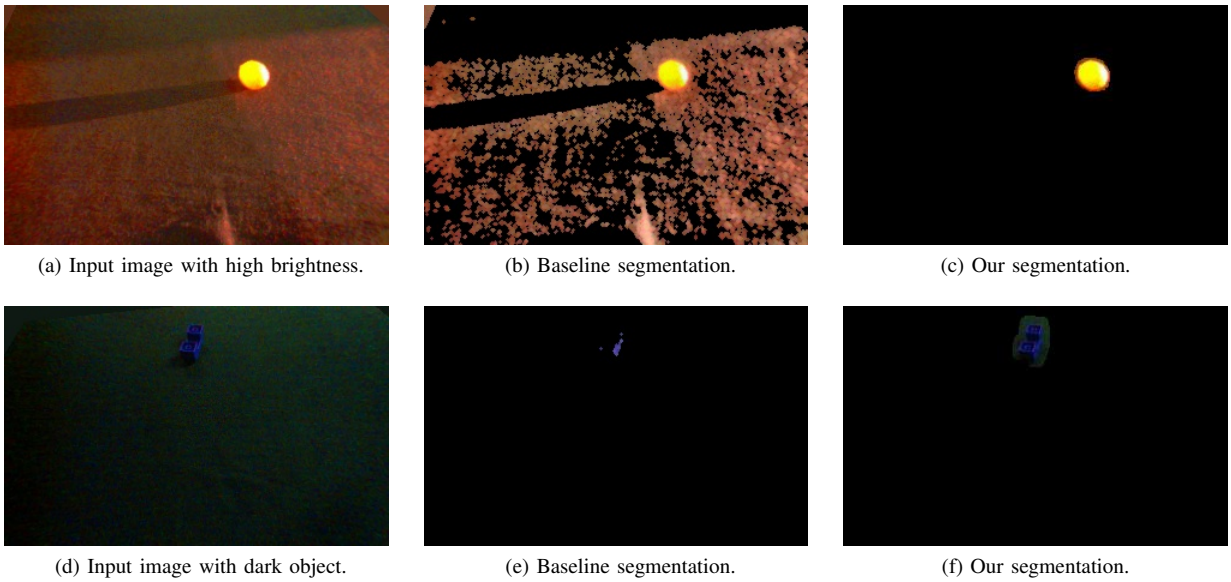
The classifier network is trained for 2000 epochs, with mini-batches of size 200. We have used Rectified Linear Unit (ReLU) activation functions for all the layers except the classification layer. Detailed architecture and parameters of the classifier network are listed in table I.

C. Forward and backward models

The architecture of forward and backward models is the lower-left corner of the whole mode depicted in Fig. 3. After the input, there are two fully-connected layers with a dropout layer in between with a 0.7 probability of keeping activations, to mitigate over-fitting. The output of these two layers first goes through another fully connected layer, before reaching the output layer, which has the same size as the input. Except for the output, we have used ReLU activation functions for all the layers. The first 60 dimensions of the output layer, corresponding to predicting the 60 important latent random variables of images, use linear activation functions, whereas the remaining twelve, corresponding to the auto-encoded action, use *softmax* activation functions. The mini-batch size of this network is reduced to 110 samples per mini-batch.

As explained, in order to obtain a smaller model and to avoid over-fitting, we adopted a *joint* training scheme, in which the two above models share the weights of the aforementioned first two hidden layers, differing only in the last hidden layer and output layer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems



(a) Input image with high brightness. (b) Baseline segmentation. (c) Our segmentation.

(d) Input image with dark object. (e) Baseline segmentation. (f) Our segmentation.

Figure 20: Examples of images from our dataset that are challenging for object segmentation. First row: an image with strong illumination variability. Second row: an image whose colours and edges are difficult to separate from the background. In both examples, we show how our new segmentation method (see Appendix B) overcomes a baseline segmentation algorithm that relies on background subtraction and intensity thresholding.

REFERENCES

[1] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat, "State representation learning for control: An overview", *Neural Networks*, vol. 108, pp. 379–392, 2018.

[2] P. R. Davidson and D. M. Wolpert, "Widespread access to predictive models in the motor system: A short review", *Journal of Neural Engineering*, vol. 2, no. 3, S313, 2005.

[3] A. Dehban, C. Cardoso, P. Vicente, A. Bernardino, and J. Santos-Victor, "Robotic interactive physics parameters estimator (rippe)", in *IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 2019, pp. 48–53.

[4] S. Zhu, A. Kimmel, K. E. Bekris, and A. Boularias, "Fast model identification via physics engines for data-efficient policy search", in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, AAAI Press, 2018, pp. 3249–3256.

[5] M. K. Kaiser, D. R. Proffitt, and M. McCloskey, "The development of beliefs about falling objects", *Perception & Psychophysics*, vol. 38, no. 6, pp. 533–539, 1985.

[6] B. Smith and R. Casati, "Naive physics", *Philosophical Psychology*, vol. 7, no. 2, pp. 227–247, 1994.

[7] R. Baillargeon, "Infants' physical world", *Current directions in psychological science*, vol. 13, no. 3, pp. 89–94, 2004.

[8] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application", in *IEEE International Joint Conference on Neural Networks*, 2012, pp. 1–8.

[9] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "Learning deep dynamical models from image pixels", *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1059–1064, 2015, 17th IFAC Symposium on System Identification SYSID.

[10] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep Spatial Autoencoders for Visuomotor Learning", in *IEEE International Conference on Robotics and Automation*, 2016.

[11] R. Jonschkowski and O. Brock, "State Representation Learning in Robotics: Using Prior Knowledge about Physical Interaction", in *Robotics: Science and Systems*, 2014.

[12] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised Learning for Physical Interaction through Video Prediction", in *Conference on Neural Information Processing Systems*, 2016, pp. 64–72.

[13] M. S. Nunes, A. Dehban, P. Moreno, and J. Santos-Victor, "Action-conditioned benchmarking of robotic video prediction models: A comparative study", *arXiv preprint arXiv:1910.02564*, 2019.

[14] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "Robonet: Large-scale multi-robot learning", in *CoRL 2019: Volume 100 Proceedings of Machine Learning Research*, 2019. arXiv: 1910.11215 [cs.RO].

[15] J. J. Gibson, *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[16] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor coordination to imitation", *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[17] E. Jaramillo-Cabrera, E. F. Morales, and J. Martinez-Carranza, "Enhancing object, action, and effect recognition using probabilistic affordances", *Adaptive Behavior*, pp. 1–12, 2019.

[18] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor, "Affordances in psychology, neuroscience and robotics: A survey", *IEEE Transactions on Cognitive and Developmental Systems*, 2016.

[19] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "Denoising auto-encoders for learning of objects and tools affordances in continuous space", in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4866–4871.

[20] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "A deep probabilistic framework for heterogeneous self-supervised learning of affordances", in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 476–483.

[21] R. Mottaghi, M. Rastegari, A. Gupta, and A. Farhadi, ""What Happens If..." Learning to Predict the Effect of Forces in Images", in *European Conference on Computer Vision*, 2016.

[22] A. Ahmetoglu, M. Y. Seker, A. Sayin, S. Bugur, J. Piater, E. Oztop, and E. Ugur, "Deepsym: Deep symbol generation and rule learning from unsupervised continuous robot interaction for planning", *arXiv preprint arXiv:2012.02532*, 2020.

[23] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine, "Learning to Poke by Poking: Experiential Learning of Intuitive Physics", in *Conference on Neural Information Processing Systems*, 2016, pp. 5074–5082.

[24] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution", in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 2450–2462.

[25] A. Antunes, G. Saponaro, A. Morse, L. Jamone, J. Santos-Victor, and A. Cangelosi, "Learn, plan, remember: A developmental robot architecture for task solving", in *IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 2017, pp. 283–289.

[26] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging inference networks and posterior collapse in variational autoencoders", in *International Conference on Learning Representations*, 2019.

[27] A. Dehban, L. Jamone, A. R. Kampff, and J. Santos-Victor, "A Moderately Large Size Dataset to Learn Visual Affordances of Objects and Tools Using iCub Humanoid Robot", in *European Conference on Computer Vision*, ser. Workshop on Action and Anticipation for Visual Learning, 2016.

[28] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[29] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols", *IEEE Robotics and Automation Magazine*, 2015.

[30] M. Zambelli, A. Cully, and Y. Demiris, "Multimodal representation models for prediction and control from partial information", *Robotics and Autonomous Systems*, 2020.

[31] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends", *IEEE signal processing magazine*, pp. 96–108, 2017.

[32] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic Backpropagation and Approximate Inference in Deep Generative Models", in *International Conference on Machine Learning*, 2014.

[33] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", in *International Conference on Learning Representations*, 2014.

[34] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance Weighted Autoencoders", in *International Conference on Learning Representations*, 2016.

[35] I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early Visual Concept Learning with Unsupervised Deep Learning", *arXiv preprint arXiv:1606.05579*, 2016.

[36] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, *Variational lossy autoencoder*, 2017.

[37] S. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space.", in *Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL).*, 2016.

[38] P. Dollár and C. L. Zitnick, "Fast Edge Detection Using Structured Forests", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1558–1570, 2015.

[39] C. L. Zitnick and P. Dollár, "Edge Boxes: Locating Object Proposals from Edges", in *European Conference on Computer Vision*, 2014.

[40] P. Wayalun, P. Chomphuwiset, N. Laopracha, and P. Wanchanthuek, "Images Enhancement of G-band Chromosome Using histogram equalization, Otsu thresholding, morphological dilation and flood fill techniques", in *IEEE International Conference on Computing and Networking Technology*, 2012, pp. 163–168.

[41] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, PMLR, 2010, pp. 249–256.

[42] G. Andrew, R. Arora, J. A. Bilmes, and K. Livescu, "Deep Canonical Correlation Analysis", in *International Conference on Machine Learning*, 2013, pp. 1247–1255.

[43] L. Pinto and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning", in *IEEE International Conference on Robotics and Automation*, 2017, pp. 2161–2168.

[44] A. Bozkurt, B. Esmaeili, D. H. Brooks, J. G. Dy, and J.-W. van de Meent, *Evaluating combinatorial generalization in variational autoencoders*, 2019. arXiv: 1911.04594 [cs.LG].

[45] D. L. Massart, L. Kaufman, P. J. Rousseeuw, and A. Leroy, "Least median of squares: A robust method for outlier and model error detection in regression and calibration", *Analytica Chimica Acta*, vol. 187, pp. 171–179, 1986.

[46] D. Pierce and B. J. Kuipers, "Map learning with uninterpreted sensors and effectors", *Artificial Intelligence*, vol. 92, no. 1, pp. 169–227, 1997.

[47] D. C. Dennett, "Current issues in the philosophy of mind", *American Philosophical Quarterly*, vol. 15, no. 4, pp. 249–261, 1978.

[48] P. Zech, E. Renaudo, S. Haller, X. Zhang, and J. Piater, "Action representations in robotics: A taxonomy and systematic classification", *The International Journal of Robotics Research*, pp. 518–562, 2019.

[49] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep Convolutional Inverse Graphics Network", in *Conference on Neural Information Processing Systems*, 2015, pp. 2539–2547.

[50] Z. Li and D. Hoiem, "Learning Without Forgetting", in *European Conference on Computer Vision*, Springer, 2016, pp. 614–629.

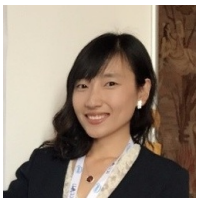
[51] L. Jamone, A. Bernardino, and J. Santos-Victor, "Benchmarking the Grasping Capabilities of the iCub Hand with the YCB Object and Model Set", *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 288–294, 2016.

[52] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)", in *International Conference on Learning Representations*, 2016.

[53] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", in *International Conference on Learning Representations*, 2015.

Atabak Dehban received the B.Sc. degree and the M.Sc. degree in computer engineering–control systems from the Amirkabir University of Technology (AUoT), Tehran, Iran, in 2012 and 2014, respectively. He finished his Ph.D. degree (Cum Laude) in 2021 at the Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal through Robotics, Brain, and Cognition (RB-Cog) doctoral programme.

Currently, he is a postdoctoral researcher in the Computer and Robot Vision Laboratory, Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa. His current research interests include affordances and visual scene understanding, self-supervised robotic machine-learning algorithms, and computer vision.



Shanghang Zhang is a Tenure Track Assistant Professor at the Computer Science Department of Peking University. She has been the postdoc research fellow at Berkeley AI Research Lab (BAIR), EECS, UC Berkeley. Dr Zhang received her Ph.D. from Carnegie Mellon University in 2018, and her Master's from Peking University. Her recent work "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting" has received the AAAI 2021 Best Paper Award. She is especially interested in machine learning with limited training data, including domain adaptation, meta-learning, and low-shot learning

Nino Cauli is a researcher at the Department of Mathematics and Computer Science of the University of Catania. He received his M.Sc. in Computer Science from the University of Pisa in 2010 (110/110), and his Ph.D. in Biorobotics from the BioRobotics Institute of Scuola Superiore Sant'Anna, Pisa, in 2014 (Cum Laude).

He has collaborated in several EU Projects such as RoboSom and Human Brain Project and his current research interests are in the areas of deep neural networks, machine learning, computer vision, internal models, predictive controllers and bioinspired robotics.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2022.3152383, IEEE Transactions on Cognitive and Developmental Systems

Lorenzo Jamone received the M.Sc. degree (with honours) in computer engineering from the University of Genoa, Genoa, Italy, in 2006, and the Ph.D. degree in humanoid technologies from the University of Genoa and Istituto Italiano di Tecnologia in 2010. He is a Lecturer in robotics with the Queen Mary University of London, London, U.K. He was an Associate Researcher with the Takanishi Laboratory, Waseda University, Tokyo, Japan, from 2010 to 2012, and with VisLab, Instituto Superior Técnico, Lisbon, Portugal, from 2012 to 2016. His current research interests include cognitive humanoid robots, motor learning and control, and force and tactile sensing.

José Santos-Victor is a Full Professor of computer vision and robotics, Instituto Superior Técnico, Lisbon, Portugal. He is the President of the Institute for Systems and Robotics, Lisbon, and the Head of the Computer and Robot Vision Laboratory, Lisbon. He is the Scientist responsible for IST in many European and national research projects in cognitive and bio-inspired computer vision and robotics. He has published over 300 articles and has an H-index of 48 in international journals and conferences.