Learning sparse and meaningful representations through embodiment

Viviane Clay a,*, Peter König a,b,1, Kai-Uwe Kühnberger a,1, Gordon Pipa a,1

a Institute of Cognitive Science, University of Osnabrück, Wachsbleiche 27, 49090 Osnabrück, Germany
b Institute of Neurophysiology and Pathophysiology, University Medical Center Hamburg-Eppendorf, Hamburg, Germany

1. Introduction

When the way supervised neural networks learn is compared to the way humans learn one can easily make out some major differences. Two of these differences are supervision and embodiment. Taking the example of object recognition from visual observations, a neural network will be presented with thousands of images of the object in question, each of them accompanied by a class label. A toddler in comparison will also collect many observations of the object of interest, however, will do so by interacting with the object, looking at it from different perspectives by moving the head or even moving the object (Bambach et al., 2018). This law-governed change in observations conditioned on the movements of the toddler emphasizes the importance of embodied cognition (Engel et al., 2013). It will make it possible to recognize the object as a distinct entity, separate from its surroundings and learn a general concept of it. This allows it to robustly recognize the object again even when seen from new perspectives or under different lighting conditions (Smith & Sline, 2017). When the toddler is now told the name of the object, an almost instantaneous association between label and object can be made without the need of thousands of labeled examples (Samuelson & Smith, 2005). This therefore makes a very efficient strategy for learning stable representations of objects.

Fully supervised neural networks have been shown to suffer from shortcomings that humans usually do not exhibit. Szegedy et al. (2013) showed how very small perturbations to an image, undetectable to the human eye, can drastically change the classification accuracy of a neural network. Even simply holding such adversarial examples in front of a camera (Kurakin et al., 2016) or specific natural images without any perturbations (Hendrycks et al., 2019) can have this effect. The networks seem to possess an over-reliance on local image features such as texture and do not consider global features such as the overall shape and outline of an object (Baker et al., 2018). Considering the training circumstances, this effect is unsurprising. The networks are expected to learn the concept of objects solely from pixel values. Without being able to interact with objects or even just looking at them from slightly different perspectives, it is very difficult to extract basic knowledge such as object and background relationships. We expect that an active exploration of the world would make it possible to learn a more general and robust concept of objects.

Already early on O’Regan and Noé (2001a) argued that even though it is clear that action requires perception, this relation also reverses. Perception and the understanding of what is perceived requires action (Noé, 2005). According to O’Regan and Noé (2001b), “experience is not something that happens in us but is something we do” (p. 99). They argue that an important part of perception is to learn how actions affect sensations. These sensory motor contingencies help us make sense of our perceptions, predict them and efficiently sample the environment for information (Engel et al., 2013).

In humans, perception is hugely influenced by how we interact with the world (Witt, 2011). Goals and the expected cost to perform actions to achieve a goal influence our perception of physical entities (Proffitt, 2006). Also, more abstract processes...
Fig. 1. (Left) Network structure of the deep reinforcement learning agent. The environment and everything coming from it is marked in yellow. The deep neural network, optimized with gradient descent, facilitated by PPO is shown in blue. Parts of the PPO algorithm used to optimize the neural network are depicted in orange. (Right) Agent observations of one frame. One can see a green door to the right of the agent. The agent receives visual and auxiliary vector observations at each frame. Visual observations are of size $168 \times 168 \times 3$. Auxiliary observations are of size 8. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

such as language comprehension are linked to action systems in the brain (Pulvermüller and Fadiga, 2010). We therefore postulate that in order to teach an artificial agent a true understanding of its (simulated) world it needs to be able to interact with the world. This paper will present results from an embodied agent acting in a virtual 3D world and learning an internal representation of its sensory input. When talking about embodiment, we are not talking about physical embodiment but rather a simulated agent, consisting of sensors, motor functions and a “brain” in an environment where its actions influence the next observations which in turn influence the following actions. The framework of learning by interacting with the world through a simulated body produced a meaningful and action-oriented internal representation of the agent’s observations, even though no semantic labels were provided.

2. Related work

There is a strong research interest in learning visual structure in an unsupervised way which can for example be approached by using auto-encoders (Tschannen et al., 2018). In its simplest form, relevant structure is supposed to change slowly (Körding & König, 2001; Wiskott & Sejnowski, 2002) facilitating learning of invariant representations. To further incorporate a time component and learn visual structure and changes over time, future frame prediction is a commonly used task (Denton & Birodkar, 2017; Finn et al., 2016; Mahjourian et al., 2016; Oliu et al., 2017; Patraucean et al., 2015; Srivastava et al., 2015; Villegas et al., 2017). However, only a few of the papers dealing with time series prediction investigate the learned representations in the network (Lotter et al., 2015; Qiao et al., 2018).

An alternative concept for unsupervised representation learning is the use of predictive coding (Rao & Ballard, 1999) which can be applied to train ANNs (Han et al., 2018; Lotter et al., 2016; van den Oord et al., 2018; Wen et al., 2018). The idea of predicting future observations based on current actions can also be used in a reinforcement learning setting to inject agents with some sense of curiosity (Pathak et al., 2017). Ha and Schmidhuber (2018) have shown that training a recurrent world model using a variational auto-encoder can increase the performance of agents in several games. Chaplot et al. (2019) show how jointly training semantic goal navigation and embodied question answering can improve performance on both tasks. Also, simply seeing a visual scene from different angles can get a network to learn disentangled representations of individual objects (Eslami et al., 2018) and get it to imagine the scene from a previously unseen viewpoint (Eslami et al., 2018; Rosenbaum et al., 2018).

Researchers who investigate representation learning in reinforcement learning agents often use additional regularization or losses to enforce a certain representation in the latent space (de Bruin et al., 2018; Lesort et al., 2018; Nachum et al., 2018). Shang et al. (2019) even get agents to explicitly learn world graphs. As such explicit constraints are biologically implausible, we will investigate what kind of representations arise naturally within an embodied training setup. Lillicrap et al. (2015) have already shown results from a simple deep reinforcement learning agent which indicate that perceptually similar observations are mapped close to each other in the latent space. We will further investigate this and look explicitly at the type of representation encoding that is learned as well as the meaningfulness of the representation and the type of information that is encoded.

3. Experiments

3.1. Training a deep reinforcement learning agent

The representation under investigation in this paper is the activation in the hidden layer of a deep neural network trained in a reinforcement learning environment. Fig. 1 shows the network structure of the agent. As input, it receives visual and auxiliary environment observations from the simulated world with size $168 \times 168 \times 3$ and size 8 respectively. The visual and auxiliary observations are first processed separately by two convolutional layers and two dense layers for the visual input and two dense layers for the auxiliary vector input, until they are concatenated into one encoded state. This encoded state has dimensionality 512 where 256 of its activations come from the visual encoding pathway and 256 from the auxiliary vector encoding pathway. The activation function used here is Swish (Ramachandran et al., 2017) which means that values can range from $-0.28$ to infinity.\(^2\) The encoded state of the high dimensional visual input and its properties will be the main focus of this paper.

Based on the encoded state, two dense layers output action probabilities and a value estimate. The action probabilities are translated into actions and executed in the environment to obtain the next set of observations. They are also used, together with the value estimate and the reward received from the environment, to optimize the neural network and thereby, train the agent to perform better actions. Proximal policy optimization (PPO), which is an efficient policy gradient method for deep reinforcement learning (Schulman et al., 2017), is applied to train the artificial neural network of the agent. More specifically, the PPO implementation and reinforcement learning framework of the Unity ML-Agents toolkit are used (Juliani et al., 2018). The overall
setup and training procedure are chosen such that an artificial neural network is trained in an embodied way with a closed loop between perception and action. The network learns by interacting with the world through a simulated body which moves around freely in the environment using its action space, constrained by physical properties of the body.

As setting for training, the environment proposed in the Unity obstacle tower challenge (Juliáni et al., 2019) is used. This environment was introduced as a new benchmark in reinforcement learning for pixel-based learning in a procedurally generated 3D environment using a sparse reward signal. The agent needs to learn to navigate through a 3-dimensional maze environment, solving successively harder tasks. Every level of the tower consists of several rooms, connected by doors. When reaching the final door on a floor the agent receives a reward of one and is placed on the next, randomly generated floor of the tower. Starting from level five the agent needs to learn to pick up a key which unlocks a key door and gives the agent access to further rooms that lead to the next level door. The key can be placed in one of the rooms on the ground or on a static or moving platform. Starting from level ten a new type of door is introduced which opens only after a puzzle is solved. The puzzle requires the agent to push a block onto a colored spot on the ground. The randomly generated floors can be illuminated in different color variations and the visual theme of the environment can vary. As overfitting to specific color values in the input or floor layouts is not useful, this makes it important for the agent to learn general and stable representations.

The agent observations are collected from a third person view RGB camera. Additionally, a small vector of auxiliary information is provided indicating the number of keys the agent is holding as well as the time remaining and the current level. Rewards are very sparse as a reward (of value 1) is only received when walking through a final level door or when picking up a key. A small reward of value 0.1 is given when walking through normal doors and when picking up small blue orbs that provide additional time. One episode ends when the agent runs out of time, which means that the better the agent gets the longer exploring the tower is possible as more time-orbs can be collected, and extra time received by going through level doors. The actions of the agent are discrete and divided into four action branches: One for moving forward or backwards, one to control the camera rotation, one for jumping and one for moving left or right. The distribution of these actions in a trained agent is shown in the Appendix (Fig. A.16).

3.2. Agent performance

After training the agent for 30 million steps using the parameters specified in Appendix A.1, level 8 is reached on average. This performance is common to agents with no memory, trained using only reinforcement learning on the obstacle tower challenge (Pleines et al., 2020). As can be expected from the network structure, the agent never exceeds level ten. The network structure used here incorporates no concept of time such that the agent is unable to solve the puzzles introduced at level ten. The puzzles require some more elaborate planning of long action sequences and following through with them even when the goal is out of sight.

Fig. 2 shows the agent performance during one inference run. This particular run lasted 4000 frames which means the agent saw 4000 observations and performed 4000 actions. As the rewards are very sparse, the agent only received a reward (1 or 0.1) in 76 of these frames which is about 1.5% of all frames (only 15 of those are the full reward of one, the other 61 frames contain a 0.1 reward). One can see how the value estimate, which expresses the reward the agent expects in the future, drops off significantly after the agent reaches level ten as the agent does not expect to solve the puzzle and receive any more rewards. One can also observe how the value estimate rises in the frames leading up to the agent entering a new level. This indicates that the agent recognizes the door to the next level and already anticipates the upcoming reward.

The results of the network trained in the embodied setting will be compared to a network trained in an unsupervised way in form of an autoencoder as well as a supervised network trained on an object classification task. The autoencoder as well as the classifier has the same network structure as the embodied agent for encoding the visual observations (see Appendix, Figs. A.10 and A.12). The observations used to train them are automatically collected by the trained agent. The autoencoder is trained on a classic frame reconstruction task where its loss is the difference between the network input and output. This training setup separates the factor of embodiment in the training without introducing semantic labels. In Fig. A.11 some example input, outputs, and activations in the encoding layer are shown. The classifier is trained on automatically labeled frames (cross-checked for correctness by humans) and outputs the image content in 4 classification branches. The first branch determines the type of door present in the frame (none, level door, normal door, key door, other door). The second, third and fourth branches determine the presence of a key, orb, or puzzle object respectively. Overall, this makes eleven output neurons which is the same output dimensionality as the actions of the embodied agent. Therefore the classifier is trained to extract compact and meaningful information about the high dimensional observations, similar to the embodied agent but in a fully supervised manner. When comparing the hidden layer activations of the three networks the same 4000 observations of one agent run are used.

It is important to note that reinforcement learning agents can have a big variability in performance between different training runs (Clary et al., 2019) even when trained with the same hyperparameters. Therefore it is recommended to present results over multiple runs to account for this variability (Colas et al., 2018; Henderson et al., 2019). In the environment presented here this was computationally infeasible. We are not introducing a new algorithm or claiming superiority in performance. Our aim is merely to look at the general structure of the representations learned.
3.3. Sparse activation patterns

We will take a look at how the activations in the hidden layer of the agent network look like. Fig. 3 shows how many of the 4000 frames of one run each of the 256 neurons in the visual encoding were active. We define a neuron as active if its activity is above zero. Since the Swish activation function allows activations to be slightly below zero, units which are called inactive here can still have small inhibitory effects on the output.

Fig. 3. Activation pattern comparison between an embodied agent (top), autoencoder (middle) and classifier (bottom). Colors indicate the percentage of frames in which the neuron in the encoded state is active (activation > 0). For visualization purposes, the 256 × 1 vector was reshaped into a grid pattern. The location of a neuron in this grid has no further meaning. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In the visual encoding of the embodied agent only an average of 4.64% (mean = 11.88, min = 3 max = 35 var = 17.89) of the neurons are activated in each frame, making it a very sparse activation pattern. Over the investigated total time, 173 of the 256 neurons (= 67.58%) are active in at least 1 frame, but only 7 of them (= 2.73%) are active in more than 40% of the frames. The most active neuron of the visual embedding is active in 74.15% of all frames. This shows that there is a wide variety in the activation patterns. The agent utilizes most but not all the available neurons in the visual embedding at some point but only activates a small subgroup in each frame.

The autoencoder has an average of 239 active neurons (93.4%) in each frame (min = 238 max = 239 var = 0.003) and 17 of the neurons are never active. In the classifier, there are no neurons that are never active and 179 neurons are on average active per frame in the visual embedding (min = 125 max = 220 var = 277.59), which is a much denser activity (69.75%). Thus, in comparison the embodied agent has learned a very sparse representation of the high dimensional image input have been shown to be more robust and stable to noise (Ahmad & Scheinkman, 2019) and has been demonstrated that sparse representations in deep reinforcement learning agents lead to better performance in several environments (Fernando Hernandez-Garcia & Sutton, 2019; Liu et al., 2019; Rafati & Noelle, 2019). It has also been argued that optimizing for the ability of an interacting system to adapt to newly intervening changes as well as for dynamical robustness leads to naturally emerging sparsity (Busiello et al., 2017).

The exact mechanism that leads to the sparse representation found here is not completely clear. As we can see a steady increase in sparsity from the beginning on (Fig. 4) it is unlikely to be caused by difficulties of solving the task or the length or training. All three networks use the same network structure leading up to the encoding as well as the same optimizer and learning rate. The difference between the three conditions lies in the output, the objective and the way the networks are optimized. Following are several control experiments narrowing down the source of sparsity in the agents representations. As a measure of sparsity we use the Gini index as suggested in Hurley and Rickard (2009) where zero corresponds to low sparsity and one to high sparsity.

Even though the classifier has a similar output dimensionality as the agent it has a different objective. Its objective is to detect objects and not to predict actions. In Fig. 5 one can see this strategy to efficiently encode input without any explicit regularization being applied. Even though there is no cost associated to using more neurons than needed to encode information, the agent learns to use sparse activation patterns and even leaves some of the available neurons completely unused. Sparse representation of the high dimensional image input have been shown to be more robust and stable to noise (Ahmad & Scheinkman, 2019) and has been demonstrated that sparse representations in deep reinforcement learning agents lead to better performance in several environments (Fernando Hernandez-Garcia & Sutton, 2019; Liu et al., 2019; Rafati & Noelle, 2019). It has also been argued that optimizing for the ability of an interacting system to adapt to newly intervening changes as well as for dynamical robustness leads to naturally emerging sparsity (Busiello et al., 2017).

The exact mechanism that leads to the sparse representation found here is not completely clear. As we can see a steady increase in sparsity from the beginning on (Fig. 4) it is unlikely to be caused by difficulties of solving the task or the length or training. All three networks use the same network structure leading up to the encoding as well as the same optimizer and learning rate. The difference between the three conditions lies in the output, the objective and the way the networks are optimized. Following are several control experiments narrowing down the source of sparsity in the agents representations. As a measure of sparsity we use the Gini index as suggested in Hurley and Rickard (2009) where zero corresponds to low sparsity and one to high sparsity.

Even though the classifier has a similar output dimensionality as the agent it has a different objective. Its objective is to detect objects and not to predict actions. In Fig. 5 one can see
Fig. 6. Sparsity over training measured with Gini index. In light blue the Gini index of several control conditions is compared to the classifier (pink) and the embodied agent (dark blue). (A and B) The agent is trained with dense, more informative rewards. Rewards are provided by a teacher agent which has already been trained in the obstacle tower. A reward of 0.00025 is received for each action branch that outputs the same action the teacher agent would take (max reward per frame: 0.001). In A the agent additionally receives the sparse rewards from the environment for events like entering the next level or picking up time orbs. (C) The agent is trained on only the first level. (D) The agent is trained on observations collected with a random policy. (E) The agent is trained with two separate representations for policy and value estimate respectively. (F) The agent is trained without the entropy over action probabilities being factored into the loss. Usually this is done with the factor beta = 0.005. Variability between multiple runs under the same conditions does exist as can be seen in the dark blue curves. The five dark blue lines correspond to five trainings of the standard embodied agent under the same conditions. Caution is needed when interpreting the exact shape of the curves as no variance of the light blue curves is shown. The overall trend of sparsification persists across runs and in spite of variability in sparsification onset and slope the conditions shown here can be excluded as possible explanations.

a network trained to predict actions in a fully supervised setup (labels provided by a trained agent). Similar to the classifier it does not learn a representation as sparse as the agents’. However, when adding more supervision in the form of dense rewards provided by a trained agent into the PPO optimization (Fig. 6A and B) the representations still turn out to be sparse. This means that the sparsification happens within the PPO algorithm or the interaction with the world and is not caused by the task of producing actions or by sparse rewards.

It could be that the shift in input statistics or environment dynamics (introducing new objects and tasks) as the agent improves its policy leads to sparsification of the network activations. To test this we train an agent in a world with only one level and one agent who collects his observations with a random policy (Figs. 6C and D). One can see that the representations also get sparser in both control conditions. Therefore, the dynamically changing properties of the environment are unlikely the cause for sparsification.

Another reason could be the simultaneous optimization of two objectives (action policy and value estimate). As both objectives are optimized on the same representation this could lead to competing updates and sparsification. However, when using two separate representations to output and optimize the policy and the value estimate the encodings also sparsify over training (Fig. 6E).

Considering the results from the control experiments in Fig. 6, a possible explanation could be the uninformative nature of the rewards and the resulting noisy gradients. The very weakly supervised nature of the embodied agent is one of its main distinguishing factors from the classifier and autoencoder. Even when using a dense and more informative reward as in 6A and B the rewards are not differentiable and can therefore not simply be used to calculate an error for directed weight update as it is done in the other two networks. The loss in the agent does not contain direct information about the difference between its prediction and the ideal output. If an action was bad, no information is given about which action should have been taken instead. The autoencoder in contrast knows exactly by how much each of its pixel outputs was off from the desired output and the classifier can calculate the cross entropy between its class predictions and the labels. This leads to both networks being able to perform much more informed and directed weight updates. How the agent in turn learns from its policy gradients and if this leads to sparsification of its representation is out of the scope of this paper. We can see that overall, the embodied training setup does not only resemble nature closer in the way we think learning is achieved, but the
Fig. 7. Correlation between the cluster assignment of frames and corresponding action combinations. Clustering performed on the visual embedding of an embodied agent (left), an autoencoder (middle) and a classifier (right). (Top) Example of correlations for action combination ‘Forward + Turn Right’ with clusters within a 20-frame window. (Bottom) Bar height represents the average correlation in a 5-frame window around 0 in the top plots for all actions and clusters. It shows the most distinct correlations between clusters and actions in the embodied agent.

Table 1
Average distance and variance within points belonging to the same action combination divided by the overall distance or variance between all encodings respectively.

<table>
<thead>
<tr>
<th>Relative distances and variances</th>
<th>Embodied agent</th>
<th>Autoencoder</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>0.96</td>
<td>1.04</td>
<td>1.02</td>
</tr>
<tr>
<td>Variance</td>
<td>0.94</td>
<td>1.09</td>
<td>1.02</td>
</tr>
<tr>
<td>Action 1–4</td>
<td>0.78</td>
<td>1.02</td>
<td>1.04</td>
</tr>
<tr>
<td>Action 5–6</td>
<td>1.32</td>
<td>1.09</td>
<td>1.24</td>
</tr>
<tr>
<td>DCI/CCI</td>
<td>0.32</td>
<td>0.02</td>
<td>0.04</td>
</tr>
</tbody>
</table>

learned representations of the visual input are also most like those found in animals.

3.4. Distinct activation patterns

To find out if we can discover a general meaningful structure in the activations of the hidden layer, we first perform k-means clustering on the activation patterns. The time series of 4000 data points, each a 256 dimensional vector, are clustered into six clusters. 4

Ideally these clusters should group the encodings into meaningful and distinct classes. In order to test this, we now correlate the six clusters with the six most common action combinations. 5

As actions and their visual execution do not always match up exactly (i.e. after pressing the jump button the agent is in the air for several frames and only reaches the highest point several frames after the action was selected) we perform the correlations for a 20-frame window. The vectors which are being correlated are both binary. For the six clusters, the binary vector encodes for each frame if it belongs to a specific cluster and for the actions the vector encodes for each frame if it belongs to a specific action combination (1) or not (0). This means that the correlation values represent the correlation of two binary vectors of length 4000. For the offset correlations, we shift the action vector either to the left or the right such that the cluster assignment at frame t now matches up with action at frame t + 1 or t − 1 respectively. This gives us the correlation between cluster assignments and actions and therefore informs us if there is structure in the visual encoding that correlates with the actions selected.

In Fig. 7, on the top left one can see the correlations of the action Forward + Turn Right with the encodings of the six clusters in the embodied agent. The highest magnitude of correlation here is at zero offset. However, also the cluster association of the observations a few frames before and after shows an increase in magnitude of the correlations. When comparing this with the encodings of an autoencoder or the classifier one can see that there is a clear association between the learned image encoding and the actions. With correlations between −0.36 and 0.8 the embodied agent has a much bigger range than the autoencoder (min = −0.12, max = 0.15) and the classifier (min = −0.19, max = 0.29). These stronger positive and negative correlations show that frames that are assigned to one cluster are more or less likely to be associated with a certain action. As the clusters are created and assigned based on the activations in the visual encoding, this means that there is a connection between distinct activation patterns and actions. Also, the activation patterns in the embodied agent preceding and following an action contain some information about it. The same correlation increase of clusters in an embodied agent can be seen when looking at the correlation of clusters with semantic image content such as level doors (see

---

4 The choice for the number of clusters was made after comparing the inter-cluster variance and silhouette score for different numbers of clusters (see Appendix, Fig. A.17).

5 For the selection of action combinations and their distribution see Appendix (Fig. A.16, right).
Fig. 8. Correlation of clusters with Doors. Comparison of embodied agent (max = 0.42, min = −0.11), autoencoder (max = 0.1, min = −0.08) and classifier (max = 0.82, min = −0.12).

These results show that the learned representations of the visual input encode semantically meaningful and action relevant information in the embodied agent.

The bottom part of Fig. 7 shows the average correlation in a 5-frame window (t − 2 to t + 2) for all six action combinations. One can see that every cluster has a unique combination of correlations or anti-correlations with the different action combinations. Some action combinations have a specific cluster which seems to mostly represent this action. The autoencoder and the classifier also show some correlations with the actions, but their magnitudes are lower than in the embodied agent.

When calculating the sum of squares of the correlations in each cluster the embodied agent outperforms the autoencoder and the classifier in every cluster. The overall sum of squared correlations for the embodied agent (0.95) is much higher than the one for the autoencoder (0.1) and the classifier (0.26). This shows that the learned encoding of the agent has a structure which correlates with the actions as well as the image content which is impressive given the dimensionality of the observations (∼84,000).

3.5. Conceptual similarities, generalization and robustness

To visualize the encodings and to investigate how conceptually similar inputs are represented, we project the activations in the visual part of the embedding layer into a two-dimensional space using t-SNE (Maaten & Hinton, 2008).

Fig. 9 shows the 4000 encoding activations projected into this 2D space, colored by the corresponding action combination. Additionally, all data points where the visual observation contained a level door are circled in red. This makes it possible to look at the spatial arrangement of encodings in high dimensional space with respect to semantic and action-oriented content.

Even in the two-dimensional projection of the data, one can see a very good separation between points associated with the different actions. Also, the frames showing level doors tend to be positioned close to each other within their respective action cluster even though they show doors under very different illumination conditions (see the three example pictures in the top left of Fig. 9). As the network’s task is not to recognize doors or other objects, but to navigate in a 3D world, it is important to encode the visual information in this way. A door in the right part of the frame needs to be encoded differently than a door in the center or the left part of the frame. However, two doors in the right part of the frame under different illumination should be encoded very similarly. This meaningful and action relevant way

---

6 A more detailed analysis of how well different object concepts can be extracted from the encodings can be found in Fig. A.18 in the Appendix.

7 t-SNE is an unsupervised method for dimensionality reduction which projects high-dimensional data into lower dimensional space while preserving distance relations between the points. Fig. A.21 shows that it discovers similar structure as PCA and k-means clustering on this data.

8 For an interactive visualization to explore the observations associated with each point see here.
of encoding the input can be seen in Fig. 9.\textsuperscript{9} It shows that in the visual encoding of the input conceptually similar images are positioned close to each other, giving low importance to perceptual similarities. This means that the network encodes the input in an action-oriented way (Clark, 1998) and is rather invariant towards irrelevant parts of the input such as illumination or texture.

To investigate if conceptually similar input images also lay close to each other in high dimensional space, we can calculate the inter-class distance and variance of encodings associated with the different action combinations. Table 1 lists the distances and variances between encodings belonging to the same action combination divided by the overall distances and variance in the data.

In the embodied agent, both distance and variance reduce strongly between the first four action combinations (0.78 and 0.58 respectively). In the autoencoder and the classifier, there is a smaller change in the distance or variance when comparing all data points to points belonging to one action combination and both their relative distances and variances are closer to one. The last two action combinations which represent backwards motion and all other rare action combinations actually have an increased variance in the embodied agent. This may be due to a very variable use of backwards motion when seeing possibly confusing visual input and the accumulation of multiple action combinations (also including jumping) in the last action category. However, we can see that at least for the first four action combinations the encodings of conceptually similar input frames are also closer together in the high dimensional space (256 dimensions) of the visual encoding.

Additionally, we calculated an overall cluster distance index (CDI) and cluster correlation index (CCI) for each network's activations. The CDI/CCI is defined by the difference between the mean distances/correlations between representations across cluster and the mean distances/correlations between representations within the clusters. To normalize this the result of the across correlation minus the within correlation is divided by the average correlation of all data points of the respective network activations. These CDI and CCI scores therefore determine how strongly the actions cluster the representations. When comparing the CDI and CCI scores the embodied agent outperforms the two other networks strongly indicating a very distinct action encoding in the network trained in an embodied setup.

### 4. Conclusion

The results presented in this paper show that a neural network, trained in an embodied framework, can learn stable and meaningful representations of its high dimensional input. Compared to an autoencoder as well as a classifier, the representations of the embodied agent are encoded in a sparse and efficient way and better reflect information encoding found in animals. The information encoded in the latent representation of the network is mainly action focused, but also contains general concepts of action relevant objects such as doors, disregarding irrelevant information such as illumination. Overall, these results suggest deep reinforcement learning as a promising method for investigating stable representation learning under weakly supervised conditions, similar to what is known from biological findings.

### CRediT authorship contribution statement

Viviane Clay: Conceived the idea to investigate latent representations in deep reinforcement learning agents, trained the agents, designed the experiments, analyzed the data, took the lead in writing the manuscript, discussed the experiment design and results and contributed to the final manuscript. Peter König: Discussed the experiment design and results, Contributed to the final manuscript, Supervised the project. Kai-Uwe Kühnberger: Discussed the experiment design and results, Contributed to the final manuscript, Supervised the project. Gordon Pipa: Discussed the experiment design and results, Contributed to the final manuscript, Supervised the project.

### Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

### Acknowledgment

The project was financed by the funds of a research training group provided by the Deutsche Forschungs Gesellschaft (DFG), Germany (GRK2340).

### Appendix

#### A.1. Training specifics

See Table A.2.

#### A.2. Autoencoder

See Figs. A.10 and A.11.

---

\textsuperscript{9} As a control that the structure does not simply originate from the statistics of the input images, t-SNE on the activations of the autoencoder and the classifier on the same input images are shown in Fig. A.19 in the Appendix.
A.3. Classifier


A.4. Neuron activations

See Fig. A.15.

A.5. Actions

See Fig. A.16.
Fig. A.13. (Left) Accuracy of the classifier during training, validation and on the 4000 hand labeled test frames used in this paper. (Right) Performance of the classifier. Counting how many of the four branches (door, key, orb, puzzle) are classified correctly in one frame. In 51% of the frames the classifier gets everything right, there are no frames where all branches are wrongly classified.

Fig. A.14. Classifier example input, encoding and output as well as the true label for four example frames.

Table A.3
Statistic of training images. 10% of these are used for validation. Therefore 34,990 images with labels are used for training and 3,879 for validation.

<table>
<thead>
<tr>
<th>Door</th>
<th>No door</th>
<th>Level door</th>
<th>Green door</th>
<th>Key door</th>
<th>Other door</th>
<th>Key</th>
<th>No</th>
<th>Yes</th>
<th>Orb</th>
<th>No</th>
<th>Yes</th>
<th>Puzzle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Samples</td>
<td>14,656</td>
<td>1,426</td>
<td>12,153</td>
<td>3,892</td>
<td>2,963</td>
<td>32,770</td>
<td></td>
<td></td>
<td>2,220</td>
<td></td>
<td>3,879</td>
<td>2.713</td>
</tr>
<tr>
<td>Class Weight</td>
<td>0.48</td>
<td>4.91</td>
<td>0.58</td>
<td>1.8</td>
<td>2.44</td>
<td>0.53</td>
<td></td>
<td></td>
<td>7.88</td>
<td></td>
<td>0.54</td>
<td>6.45</td>
</tr>
</tbody>
</table>

Table A.4
Classifier performance quantified in accuracy, precision and recall. Additionally, the number of true positives, true negatives, false positives and false negatives are given for the 4000-frame example run used in this paper.

<table>
<thead>
<tr>
<th>Door</th>
<th>No door</th>
<th>Level door</th>
<th>Green door</th>
<th>Key door</th>
<th>Other door</th>
<th>Key</th>
<th>Orb</th>
<th>Puzzle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>75.4%</td>
<td>96.9%</td>
<td>75.8%</td>
<td>85.4%</td>
<td>90.45%</td>
<td>97.4%</td>
<td>93.2%</td>
<td>87.6%</td>
</tr>
<tr>
<td>Precision</td>
<td>84.6%</td>
<td>62.5%</td>
<td>45.9%</td>
<td>34.5%</td>
<td>52.7%</td>
<td>5.1</td>
<td>50.8%</td>
<td>79.1%</td>
</tr>
<tr>
<td>Recall</td>
<td>64.8%</td>
<td>82%</td>
<td>55%</td>
<td>41.1%</td>
<td>75.1%</td>
<td>11.8%</td>
<td>67.9%</td>
<td>59.7%</td>
</tr>
<tr>
<td>TP, TN</td>
<td>[1355, 1661]</td>
<td>[150, 3727]</td>
<td>[486, 2545]</td>
<td>[176, 3238]</td>
<td>[311, 3307]</td>
<td>[4, 3892, 74, 30]</td>
<td>[188, 3541, 182, 89]</td>
<td>[531, 2971, 140, 358]</td>
</tr>
<tr>
<td>TP, FN</td>
<td>247, 737</td>
<td>90, 33</td>
<td>572, 397</td>
<td>334, 252</td>
<td>279, 103</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>
Fig. A.15. (top left) Distribution of activations over all neurons and all frames in the three conditions. Maximum activation is at 314.9, 3.8 and 64.5 for the three conditions respectively. (top right, middle and bottom). Comparison of single neuron activations in the hidden layer of the three networks during the test run. Neuron 202 (bottom left) is the most active neuron in the embodied agent. Solid color shows the activations smoothed with a Gaussian filter ($\sigma = 4$). Faded color shows the original activations.

A.6. K-means clustering on the embodied agent

See Fig. A.17.

A.7. Quality of object encodings — extracting concepts using SVMs

See Fig. A.18.

A.8. T-SNE

See Figs. A.19 and A.20.

A.9. T-SNE comparison to k-means and PCA

See Fig. A.21.

A.10. Analysis of high-D space

See Fig. A.22.

A.11. Robustness

See Fig. A.23.
Fig. A.16. Distribution of actions over one inference run of 4000 frames. (Left) Distribution in the four action branches. (Top Right) Distribution of all existing action combinations. (Bottom Right) Distribution of all action combinations used in the paper. All actions marked in gray in the plot above are aggregated in the last action combination.

Fig. A.17. (Top) Analysis used for the choice of number of clusters in Section 3.4. Both mean within class variance as well as the silhouette score for different number of clusters have their minimum at six clusters. (Bottom) Statistics of the k-means clustering with six clusters.
Fig. A.18. Quality of object encodings measured by the performance of a support vector machine (SVM) on the different representations over the 4000 images analyzed in this paper. The SVM was fitted on 1600 labeled images using a linear kernel for each concept. Training images were distributed as equal as possible over the concepts, given environment constraints, with 250–450 positive images per concept and every possible door/key/orb/puzzle combination being represented with 50 examples where possible. Class imbalances are compensated by class weights. Performance is measured in accuracy (top) and F1 score which takes precision and recall into account (bottom).

Fig. A.19. T-SNE performed on the visual embedding of an autoencoder (top) and a classifier (bottom) colored by action combination associated with each frame. Frames where the observation contained a level door are circled red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
Fig. A.20. T-SNE performed on the visual embedding of an embodied Agent (top), an autoencoder (middle) and a classifier (bottom) colored by semantic content of the visual observation associated with the encoding.
Fig. A.21. Comparison of different unsupervised clustering techniques. (Top) t-SNE colored by cluster assignment calculated with k-means clustering. (Bottom) Data projected on two largest principal components (direction of biggest variance) colored by action. All projection techniques find similar structure in the data.

Fig. A.22. Distances (left) and variances (right) within the encodings for individual actions in the original high dimensions encoding space (dim = 256). Data in numbers is also given in Table 1 accumulated over actions.
Fig. A.23. Susceptibility of agent, classifier and autoencoder to different amounts of Gaussian noise. (Top) Noise with difference variance applied to one example image. (Bottom) Effect of noise on the output (left) and the encoding (middle and right). As the autoencoder has continuous output values between 0 and 1, a changed output is defined as changing a pixels’ value by more than 0.01.

A.12. Generalizability

See Figs. A.24–A.27.

Fig. A.24. To test generalizability, the environment is rendered with lighting conditions that have not been present during training (see example images). The embodied agent still performs 22.9 times better than an untrained agent, reaching floor 3.2 on average (max = 9, min = 0) over 100 runs compared to floor 8.1 (max = 11, min = 2) under normal illumination. The autoencoder and classifier are 11.3 and 1.9 times better than random respectively under the unknown lighting conditions of one test run (3300 frames). The three conditions are between 1.56 (autoencoder) and 2.54 (agent) times worse on the generalization test compared to the trained image space. This result does not definitively claim one condition to generalize better than the others. Level reached (determined by a series of good actions), mean squared error of reconstruction and F1 score of predictions are three very different measures that are difficult to compare. One can say that all three conditions can generalize to a certain extent. Figs. A.25–A.27 show some more detailed results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
**Fig. A.25.** The agents encoding of the observations under new illumination conditions are similar to the encodings under the trained lighting up to level 6 (left). At level 6 of this test run the agents representations differ from the representations under training conditions (right) and the agent fails to reach the next level. Level 6 in this case has the yellow lighting shown in the last picture of the third row in Fig. A.24. A short video of the agent run can be found [here](#).

---

**Autoencoder Image Reconstruction - Trainings Conditions**

![Autoencoder Image Reconstruction - Trainings Conditions](image1)

- mse: 0.0066
- mse: 0.0072
- mse: 0.0019
- mse: 0.0025
- mse: 0.0039

---

**Autoencoder Image Reconstruction - Test Conditions**

![Autoencoder Image Reconstruction - Test Conditions](image2)

- mse: 0.0025
- mse: 0.0026
- mse: 0.0016
- mse: 0.0055
- mse: 0.0103

---

**Fig. A.26.** The autoencoder can reconstruct the general structure of the input images but often fails to capture the new color of lighting (see middle row of the test condition images).
Fig. A.27. The classifiers representation of the differently lit images is not as structured into object classes as on the training space (see A.20). Classification performance drops from an accuracy of 85.9% to 78.3% but is still better than random performance (shuffled labels) of 70.7%. The F1 score drops from 56.8 to 31.4 (random = 16.1).

References


40