EFFERENCENETS FOR LATENT SPACE PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning in high-dimensional space remains a challenging problem, even with recent advances in algorithms and computational power. We are inspired by efference copy and sensory reafference theory from neuroscience. Our aim is to allow agents to form mental models of their environments for planning. The cerebellum is emulated with a two-stream, fully connected, predictor network. The network receives as inputs the efference as well as the features of the current state. Building on insights gained from knowledge distillation methods, we choose as our features the outputs of a pre-trained network, yielding a compressed representation of the current state. The representation is chosen such that it allows for fast search using classical graph search algorithms. We display the effectiveness of our approach on a viewpoint-matching task using a modified best-first search algorithm.

1 INTRODUCTION

As we manipulate an object in our hands, we can accurately predict how it looks after some action is performed. Through our visual sensory system, we receive high-dimensional information about the object. However, we do not hallucinate its full-dimensional representation as we estimate how it would look and feel after we act. But we feel that we understood what happened if there is an agreement between the experience of the event and our predicted experience.

There has been much recent work on methods that take advantage of compact representations of states for search and exploration. One of the advantages of this approach is that finding a good representation allows for faster and more efficient planning. This holds in particular when the latent space is of a much lower dimensionality than the one where the states originally live in.

Our central nervous system (CNS) sends a command (efferent) to our motor system, as well as sending a copy of the efferent to our cerebellum, which is our key organ for predicting the sensory outcome of actions when we initiate a movement and is responsible for fine motor control. The cerebellum then compares the result of the action (sensory reafference) with the intended consequences. If they differ, then the cerebellum makes changes to its internal structure such that it does a better job next time — i.e., in no uncertain terms, it learns.

The cerebellum receives 40 times more information than it outputs, by a count of the number of axons. This gives us a sense of the scale of the compression ratio between the high dimensional input and low dimensional output. Thus, we constrain our attention to planning in a low-dimensional space, without necessarily reconstructing the high-dimensional one.

We apply this insight for reducing the complexity of tasks such that planning in high dimensionality space can be done by classical AI methods in low dimensionality space . Our **contributions** are thus twofold: provide a link between efference theory and classical planning with a simple model and introduce a search method for applying the model to reduced state-space search.

We validate our approach experimentally on visual data associated with categorical actions that connect the images, for example taking an object and rotating it. We create a simple manipulation task using the NORB dataset (LeCun et al., 2004), where the agent is presented with a starting viewpoint of an object and the task is to produce a sequence of actions such that the agent ends up with the target viewpoint of the object. As the NORB data set can be embedded on a cylinder (Schüler et al., 2018) (Hadsell et al., 2006) or a sphere (Wang et al., 2018), we can visualize the actions as traversing the embedded manifold.

2 RELATED WORK

We essentially aim at approximating a Markov decision process' state transition function. Thus, similarities abound in the reinforcement learning literature: many agents perform explicit latent-space planning computations (Tamar et al., 2016) (Srinivas et al., 2018) (Hafner et al., 2018) (Henaff et al., 2017) (Chua et al., 2018) (Gal et al., 2016) as part of learning and executing policies. Gelada et al. (2019) train a reinforcement learning (RL) agent to simultaneously predict rewards as well as future latent states. Our work is distinct from these as we are not assuming a reward signal and are not constrained to an RL setting during training.

Similar to our training setup, Oh et al. (2015) predict future frames in ATARI environments conditioned on actions. The predicted frames are used for learning the dynamics of the environment, e.g. for improving exploration by informing agents of which actions are more likely to result in unseen states. Our work differs as we are maneuvering within the latent space and not the full input space.

Vision, memory, and controller modules are combined for learning a model of the world before learning a decision model in Ha and Schmidhuber's World Models (Ha & Schmidhuber, 2018). A predictive model is trained in an unsupervised manner, allowing the agent to learn policies entirely within its learned latent space representation of the environment. Instead of training the representation from scratch, we apply the machinery of transfer learning by using pre-trained networks. This allows the method to be applied to a generic representation rather than a specifically trained representation.

Using the output of pre-trained networks as targets instead of the original pixels is not new. The case where the output of a larger model is the target for a smaller model is known as knowledge distillation (Hinton et al., 2015) (Buciluǎ et al., 2006). This is used for compressing a model ensemble into a single model. Vondrick et al. (Vondrick et al., 2016) learn to make high-level semantic predictions of future frames in video data. Given a frame at a current time, a neural network is tasked with predicting the representation of a future frame, e.g. by AlexNet. Our approach extends this general idea by admitting an action as the input to our predictor network as well.

Causal InfoGAN (Kurutach et al., 2018) is a method based on generative adversarial networks (GANs) (Goodfellow et al., 2014), inspired by InfoGAN in particular (Chen et al., 2016), for learning plannable representations. A GAN is trained for encoding start and target states and plans a trajectory in the representation space as well as reconstructing intermediate states in the plans. Our method differ from this by training a simple forward model and forgoing reconstruction, which is unnecessary for planning.

3 EFFERENCE COPIES

To motivate our approach, we will briefly describe the concept of *efference copies* from neuroscience. As we act, our central nervous system (CNS) sends a signal, or an *efference*, to our peripheral nervous system (PNS). An additional copy of the efference is created, which is sent to an internal forward model (Jeannerod & Arbib, 2003). This model makes a prediction of the sensory outcome and learns by minimizing the errors between its prediction and the actual sensory inputs that are experienced after the action is performed.

The sensory inputs that reach the CNS from the sensory receptors in the PNS are called *afference*. They can be *exafference*, signals that are caused by the environment, or *reafference*, signals that are caused by ourself acting. By creating an efference copy and training the forward model, we can tell apart exafference from reafference. This is how we can be tickled when grass straws are brushed against our feet (exafference), but we can walk barefeet over a field of grass without feeling tickled (reafference).

We assume that the motor system and sensory system are fixed and not necessarily trainable. The motor system could be the transition function of a Markov decision process. The sensory system is assumed to be a visual feature extractor — in our experiments, a Laplacian Eigenmap or an intermediate layer of a pre-trained convolutional neural network (CNN) is used. Pre-trained CNNs can provide powerful, generic descriptors (Sharif Razavian et al., 2014) while eliminating the computational load associated with predicting the pixel values of the full images (Vondrick et al., 2016).

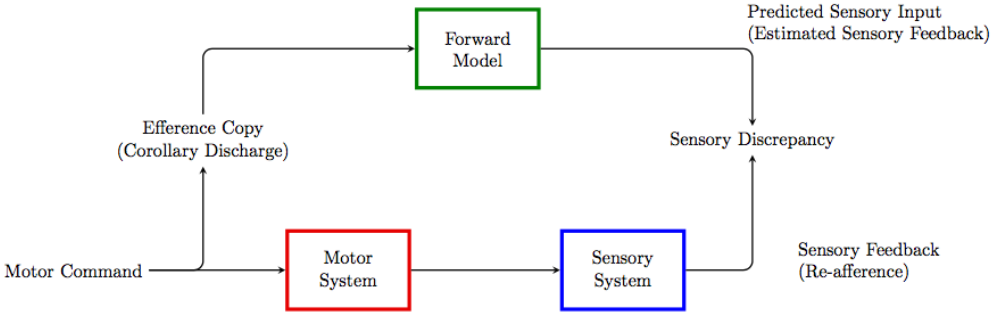

Figure 1: **Efference copy diagram** (Gwijde, 2018). A motor command, or efference, is issued by the CNS. A copy of the efference is made and is sent to an internal forward model, which predicts the sensory result of the efference. The original efference is sent to the motor system to act. The consequence of the action in the world is observed by the sensory system, and a reafference is created and sent to the CNS. The forward model is then updated to minimize the discrepancy between predicted and actual reafference.

4 Proposed Method

Suppose we have a feature map ϕ and a training set of N data points $(x_i = [\phi(S_t), a_t], y_i = \phi(S_{t+1}))$, where S_t is the state at time step t and a_t is an action resulting in a state S_{t+1}. We train the predictor f, parameterized by θ, by minimizing the mean squared error loss over f's parameters:

$$\underset{\theta}{\operatorname{argmin}}\, L(D, \theta) = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{N} \sum_{i=1}^{N} \left(\phi(S_{t+1}) - f_\theta(\phi(S_t), a_t)\right)^2 \tag{1}$$

where $D = \{(x_i, y_i)\}_{i=0}^{N}$ is our set of training data.

In our experiments, we construct f as a two-stream, fully connected, neural network (Fig. 2). Using this predictor we are able to carry out efficient planning in the latent space defined by ϕ. By planning we mean that there is a start state S_{start} and a goal state S_{goal} and we are tasked with finding a path between them.

Assuming deterministic dynamics, we output the expected representation after performing the action. This allows us to formulate planning as a classical AI pathfinding or graph traversal problem. The environment is turned to a graph by considering each state as a node and each action as an edge.

4.1 Search with EfferenceNets

We used a modified best-first search algorithm with the trained EfferenceNets for our experiments (Algorithm 1). Each state is associated with a node as well as a score: the distance between its representation and the representation of the goal state. The edges between nodes are the available actions at each state. The output of the search is the sequence of actions that corresponds to the path connecting the start node to the proposed solution node, i.e. the node whose representation is the one closest to the goal.

To make the algorithm faster, we only consider paths that do not take us to a state that has already been evaluated, even if there might be a difference in the predictions from going this roundabout way. That is, if a permutation of the actions in the next path to be considered is already in an evaluated path, it will be skipped. This is akin to transposition tables used to speed up search in game trees. This might yield paths with redundancies which can be amended with path-simplifying routines (e.g. take one step forward instead of one step left, one forward then one right).

4.2 On Suitable Representations

Selecting the optimal action in each step of a temporally-extended Markov decision process in a task effectively is a hard problem that is solved by different fields in different ways. For example, in

Algorithm 1 Find a plan to reach $S_{\text{result}} = S_{\text{j}}$, where $j = \text{argmin}_{i \leq k} ||R_i - \phi(S_{goal})||$ and $R = (r_0, \ldots, r_k)$ are the representations of explored states.

Input: S_{start}, S_{goal}, max. trials m, action set A, feature map ϕ and EfferenceNet f
Output: A plan of actions $(a_0, \ldots, a_n)$ reaching $S_{\text{result}} \approx S_{\text{goal}}$ from S_{start}

```
R ← ∅, A ← ∅, D ← ∅, p_0 ← ∅
r_0 ← φ(S_start)                                  # add j to the set of checked indices
for k ← 0 to m do
    j = argmin_{i≤k, i∉D} ||r_i − φ(S_goal)||     # take a new state, most similar to the goal
    for all a ∈ A do
        r_{k+1} ← f(r_j, a)                       # try every action from current state
        R ← R ∪ {r_{k+1}}                         # save the estimated representation
        p_{k+1} ← p_j ∪ a
        P ← P ∪ {p_{k+1}}                         # store path from goal state to current state
        k ← k + 1
    end for
    D ← D ∪ j                                     # add j to the set of checked indices
end for
return p_j
```

the area of reinforcement learning, it is addressed by estimating the accumulated reward signal for a given behavioral strategy and adapting the strategy to optimize this estimate. This requires either a large number of actively generated samples or exact knowledge of the environment (in dynamic programming) to find a strategy that behaves optimally.

In this paper, we choose a different approach and utilize a one-step prediction model to allow decisions to be made based on the predicted outcome of a number of one-step lookaheads started from an initial state. The actions are chosen so that each step greedily maximizes progress towards a known target. This method, sometimes called hillclimber or best-first search, belongs to the family of informed search algorithms (Nilsson, 2014).

To be more efficient than random or exhaustive search, these kinds of algorithms rely on heuristics to provide sufficient evidence for a good — albeit not necessarily optimal — decision at every time step to reach the goal. Here we use the Euclidean distance in representation space: An action is preferred if its predicted result is closest to the goal. The usefulness of this heuristics depends on how well and how coherently the Euclidean distance encodes the actual distance to the goal state in terms of the number of actions.

Our experiments show that an easily attainable general purpose representation, such as a pre-trained VGG16 (Simonyan & Zisserman, 2014), can already provide sufficient guidance to apply such this heuristic effectively. One might, however, ask what a particularly suited representation might look like when attainability is ignored. It would need to take the topological structure of the underlying data manifold into account, such that the Euclidean distance becomes a good proxy for the geodesic distance. One class of methods that fulfill this are **spectral embeddings**, such as Laplacian Eigenmaps (LEMs) (Belkin & Niyogi, 2003). Since they do not readily allow for out-of-sample embedding, they will only be applied in an in-sample fashion to serve as a control experiment in Section 5.2.1.

5 EXPERIMENTS

In the experiments, we show that our method can be combined with simple, informed graph search (Russell & Norvig, 2016) algorithms to plan trajectories for manipulation tasks (Fig. 3, top row). We use the NORB dataset, which contains images of different objects each under 9 different elevations, 36 azimuths, and 6 lighting conditions. We derive from the dataset an object manipulation environment. The actions correspond to turning a turntable back and forth by $20°$, moving the camera up or down by $5°$ or changing the illumination intensity.

After the EfferenceNet is trained, we apply Algorithm 1 to the viewpoint matching task. The goal is to find a sequence of actions that transforms the start state to the goal state. The two states differ in their azimuth and elevation configuration.

5.1 TRAINING AND ARCHITECTURE

Given a feature map ϕ, we task the EfferenceNet f with predicting $\phi(S_{t+1})$ after the action a was performed in the state S_t. We train f by minimizing the mean-squared error between $f(\phi(S_t), a)$ and $\phi(S_{t+1})$.

The network (Fig. 2 is built with Keras (Chollet et al., 2015) and optimized with Nadam (Dozat, 2016) and converges in two hours on a Tesla P40. It is a two-stream dense neural network. Each stream consists of one dense layer followed by a batch normalization (BatchNorm) layer. The outputs of these streams are then concatenated and passed through 3 dense layers, each one followed by a BatchNorm, and then an output dense layer. Every dense layer, except the last, is followed by a rectified linear unit (ReLU) activation.

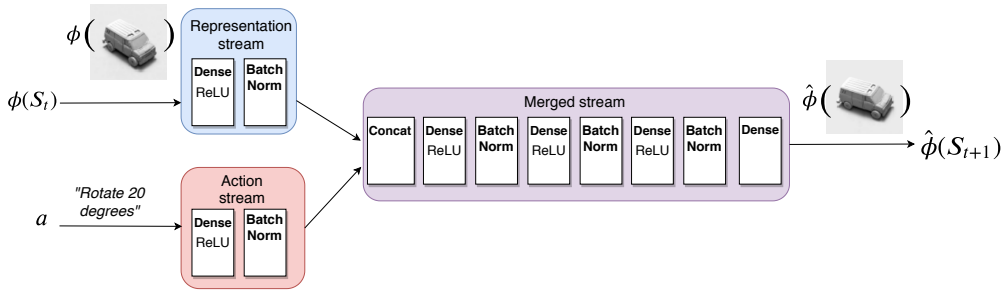

Figure 2: EfferenceNet architecture. The network takes as input the representation vector $\phi(S_t)$ of the state S_t as determined by the feature map ϕ, as well as the one-hot encoded action a. It outputs the estimated feature vector of the resulting state S_{t+1} after action a is performed.

Our ϕ is chosen to be a Laplacian Eigenmap for in-sample search and the second-to-last layer output of VGG16 (Simonyan & Zisserman, 2014), pre-trained on ImageNet (Deng et al., 2009), for out-of-sample search. As the representation made by ϕ do not change over the course of the training, they can be cached, expediting the training. Of the 10 car toys in the NORB dataset, we randomly chose 9 for our training set and test on the remaining one.

5.2 NORB VIEWPOINT MATCHING

5.2.1 IN-SAMPLE

Embedding a single toy's confgurations in three dimensions using Laplacian Eigenmaps will result in a cylindrical embedding that encodes both, elevation and azimuth angles, as visible in Figure 4. Three dimensions are needed so that the cyclic azimuth can be embedded correctly as $sin(\theta)$ and $cos(\theta)$.

If such a representation is now used to train the EfferenceNet which is subsequently applied in Algorithm 1, one would expect monotonically decreasing distance the closer the prediction comes to the target. Figure 5 shows that this is the case and that this behavior can be very effectively used for a greedy heuristic. While the monotinicity is not always exact due to imperfections in the approximate prediction, Figure 5 still qualitatively illustrates a best-case scenario.

5.2.2 OUT-OF-SAMPLE

The goal and start states are chosen randomly, with the constraint that their distances along each dimension (azimuth and elevation) are uniformly distributed. As the states are searched, a heat map of similarities is produced (Fig. 3).

To visualize the performance of the search we plot a histogram (Fig. 6) illustrating the accuracy of the search. The result looks less accurate with respect to elevation than azimuth, but this is due to the elevation changes being more fine-grained than the azimuth changes, namely by a factor of 4. The difference between the elevation of the goal and solution viewpoints in Figure 3 left, for example, is

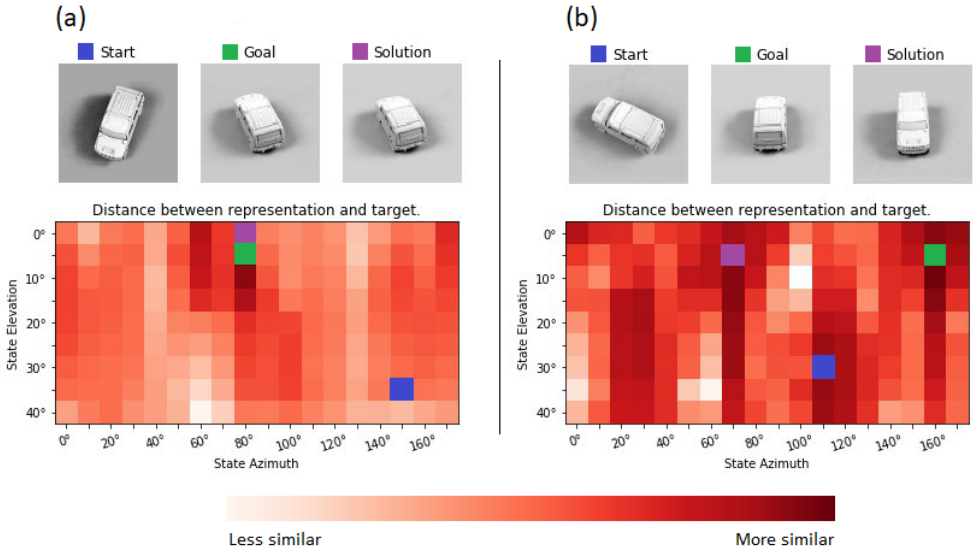

Figure 3: Viewpoint matching task on unseen objects. Only the start and goal states are given. The blue dot indicates the start state, green the goal and purple the solution state found by the algorithm. The EfferenceNet estimates the VGG16 representation of the resulting states as the object is manipulated. **(a)** The goal lies on a hill containing a maximum of representational similarity. **(b)** The accumulated noise of iterated estimations causes the algorithm to plan a path to a wrong state, albeit one with a similar shape.

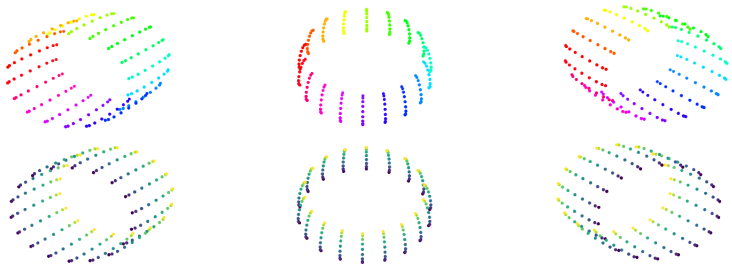

Figure 4: The three-dimensional Laplacian Eigenmaps of a toy car with azimuth (top) or elevation (bottom) colored in. Euclidean distance is a good proxy for geodesic distance in this case.

hardly perceptible. If one would scale the histograms by angle and not by bins, the drop-off would be comparable.

The heat maps of the type shown in Figure 3 can be aggregated to reveal basins of attraction during the search. Each heat map is shifted such that the goal position is at the bottom, middle row (Fig. 7, a). Here it is apparent that the goal and the $180°$ flipped (azimuth) version of the goal are attractor states. This is due to the feature map being sensitive to the rough shape of the object, but being unable to distinguish finer details. In (Fig. 7, b) we display an aggregate heat map when the agent can alter the lighting conditions as well.

5.3 Advantages and Disadvantages

In our work, we focus on learning a transition model. Doing control after learning the model is an established approach, with the mature field of model-based RL dedicated to it. This has the advantage of allowing for **quick learning of new reward functions**, since disentangling reward contingencies from the transition function is helpful when learning for multiple/changing reward functions and allows useful learning when there is no reward available at all. Thus, it might also be useful in a sparse or late reward setting.

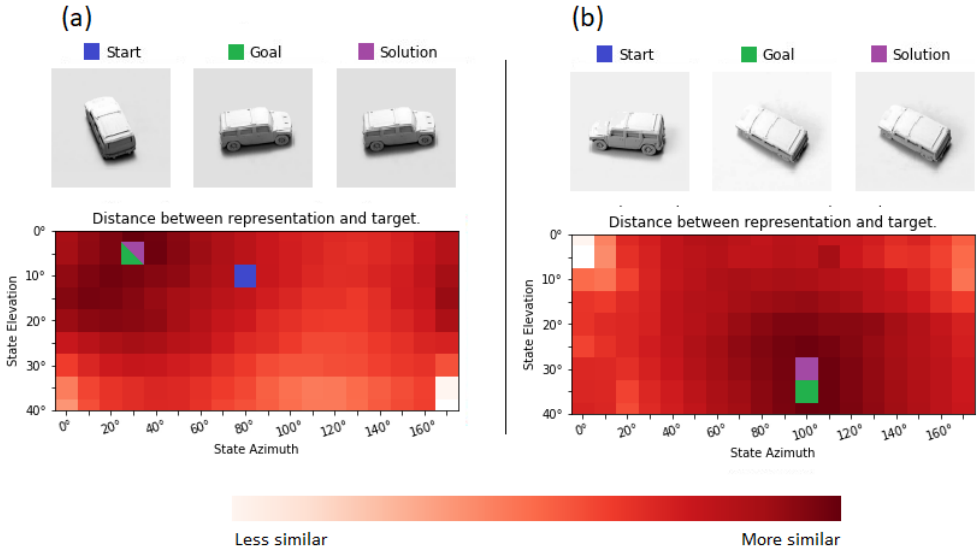

Figure 5: Viewpoint matching task as before, based on a Laplacian Eigenmap representation. The search algorithm can rely on an almost monotonously increasing similarity between prediction and target to guide its search.

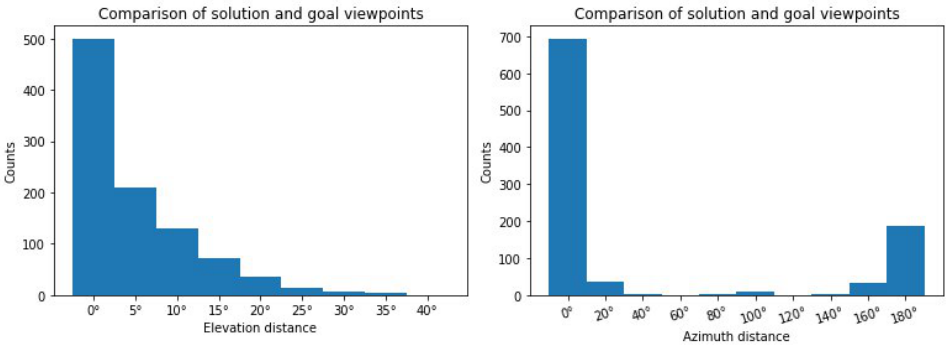

Figure 6: Histograms of distance between goal and solution states along elevation (left) and azimuth (right) on test data. The distance between the start and goal viewpoints is equally distributed across all the trials, along both dimensions. The goal and the $180°$ flipped (azimuth) version of the goal are attractor states.

Another advantage of our approach is that it accommodates **evaluations of reward trajectories** with arbitrary discounts. Standard RL methods are usually restricted in their optimization problems. Often, there is a choice between optimizing discounted or undiscounted expected returns. Simulation/rollout-based planning methods are not restricted in that sense: If you are able to predict reward trajectories, you can (greedily) optimize arbitrary functions of these — possibly allowing for behavior regularization. For example, the risk-averse portfolio manager can prioritize smooth reward trajectories over volatile ones.

We use a pre-trained network because we believe that a flexible algorithm should be based rather on generic, multi-purpose representions and not on very specific representations. This contributes to the flexibility of the system.

However, a drawback of using pre-trained networks is that features might be encoded that are irrelevant for the current task. This has the effect that informed search methods, such as best-first search, are not guaranteed to output the accurate solution in the latent space, as there might be distracting pockets of erroneous local minima. Our visualizations reveal gradient towards the goal state

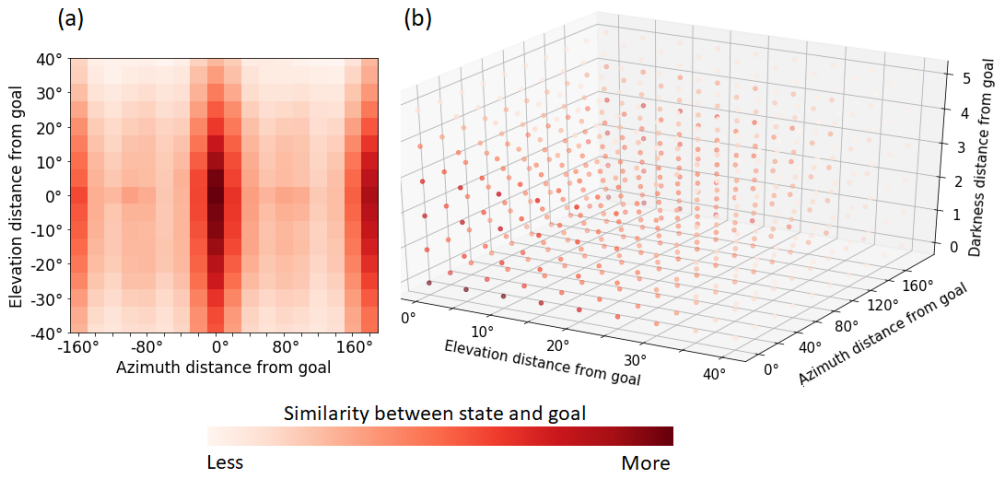

Figure 7: Aggregate heat maps of representation similarities on test data. The data is collected as the state space is searched for a matching viewpoint. The pixels are arranged according to their elevation and azimuth difference from the goal state. **(a)** We see clear gradients towards the two basins of attraction. The change is noticably less along the elevation due to less change at each step. **(b)** The agent can also alter the lighting of the scene, with qualitatively similar results. In this graphic we only measure the absolute value of the distance.

as well as a visually similar, far away states. There is variation in the similarities, preventing the planning algorithm from finding the exact goal for every task, sometimes yielding solutions that are the polar-opposites of the goal, w.r.t. the azimuth.

6 Conclusion

Pairing the EfferenceNet with a good but generic feature map allows us to perform an accurate search in the latent space of manipulating unseen objects. This remarkably simple method, inspired by the neurology of the cerebellum, reveals a promising line of future work. We validate our method by on a viewpoint-matching task derived from the NORB dataset.

In the case of deterministic environments, EfferenceNets calculate features of the current state and action, which in turn define the next state. This opens up a future direction of research by combining EfferenceNets with successor features (Barreto et al., 2017).

Furthermore, the study of effective feature maps strikes us as an important factor in this line of work to consider. We utilize here Laplacian Eigenmaps and pre-trained deep networks. It is probably possible to improve the performance of the system by end-to-end training but we believe that it is more promising to work on generic multi-purpose representations. Possible further methods include Slow Feature Analysis (SFA) (Wiskott & Sejnowski, 2002) (Schüler et al., 2018). SFA has been previously shown (Sprekeler, 2011) to solve a special case of LEMs while it allows for natural out-of-sample embeddings.

References

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541. ACM, 2006.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

François Chollet et al. Keras. `https://keras.io`, 2015.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Timothy Dozat. Incorporating Nesterov momentum into ADAM. 2016.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, 2016.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *arXiv preprint arXiv:1906.02736*, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gwijde. Efference copy, 2018. URL `https://en.wikipedia.org/wiki/File:Efference_Copy.png`. licensed under CC BY-SA 4.0: `https://creativecommons.org/licenses/by-sa/4.0/deed.en`.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Mikael Henaff, William F Whitney, and Yann LeCun. Model-based planning with discrete and continuous actions. *arXiv preprint arXiv:1705.07177*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

M Jeannerod and M Arbib. Action monitoring and forward control of movements. *The Handbook of Brain Theory and Neural Networks,*, pp. 83–85, 2003.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal InfoGAN. In *Advances in Neural Information Processing Systems*, pp. 8733–8744, 2018.

Yann LeCun, Fu Jie Huang, Leon Bottou, et al. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)*, pp. 97–104. Citeseer, 2004.

Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pp. 2863–2871, 2015.

Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

Merlin Schüler, Hlynur Davíð Hlynsson, and Laurenz Wiskott. Gradient-based training of slow feature analysis by differentiable approximate whitening. *arXiv preprint arXiv:1808.08833*, 2018.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Henning Sprekeler. On the relation of slow feature analysis and laplacian eigenmaps. *Neural computation*, 23(12):3287–3302, 2011.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 98–106, 2016.

Xiaohan Wang, Tengyu Ma, James Ainooson, Seunghwan Cha, Xiaotian Wang, Azhar Molla, and Maithilee Kunda. The toybox dataset of egocentric visual object transformations. *arXiv preprint arXiv:1806.06034*, 2018.

Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.