Unsupervised Learning of State Representations for Multiple Tasks

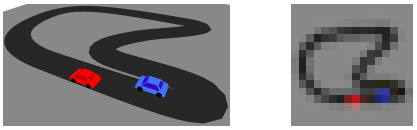
Sebastian Höfer[1,*], **Antonin Raffin**[2,*], **Rico Jonschkowski**[1], **Oliver Brock**[1], **Freek Stulp**[3]
[1] Robotics and Biology Laboratory, Technische Universität Berlin, Germany
{sebastian.hoefer,rico.jonschkowski,oliver.brock}@tu-berlin.de
[2] École Nationale Supérieure de Techniques Avancées (ENSTA-ParisTech), Paris, France
antonin.raffin@ensta-paristech.fr
[3] Robotics and Mechatronics Center, German Aerospace Center (DLR), Wessling, Germany
freek.stulp@dlr.de

Abstract

We present an approach for learning state representations in multi-task reinforcement learning. Our method learns multiple low-dimensional state representations from raw observations in an unsupervised fashion, without any knowledge of which task is executed, nor of the number of tasks involved. The method is based on a gated neural network architecture, trained with an extension of the *learning with robotic priors* objective. In simulated experiments, we show that our method is able to learn better state representations for reinforcement learning, and we analyze why and when it manages to do so.

1 Introduction and Related Work

Effective reinforcement learning hinges on an appropriate state representation for a particular task. Whereas state representations are commonly hand-crafted, novel learning methods are able to extract state representations from raw input data, for example from images (Lange et al., 2012; Mnih et al., 2015; Jonschkowski & Brock, 2015; Watter et al., 2015; Levine et al., 2015). Most of these methods focus on policy and representation learning for single tasks and therefore exhibit limited generalization capabilities.



(a) (b)

Figure 1: Slot car racing – the agent has learn how to drive any of the cars as far as possible (left), based on its raw observations (right).

In this paper, we present MT-LRP, an algorithm for learning state representations for **m**ultiple **t**asks by **l**earning with **r**obotic **p**riors. MT-LRP is able to acquire different low-dimensional state representations for multiple tasks in an unsupervised fashion. Importantly, MT-LRP does not require knowledge about which task is executed at a given time or about the number of tasks involved. This is an important requirement for robotic life-long learning, where robots should be able to determine autonomously if a task requires a separate state representation (grasping a pen is different from opening a door) or not (grasping a red or brown cup can be achieved with the same state representation). The representations learned with MT-LRP enable the use of standard reinforcement learning methods to compute effective policies from few data.

MT-LRP is implemented as two neural networks, coupled by a *gating* mechanism (Sigaud et al., 2015) as illustrated in Figure 2. The first network, χ, detects which task is being executed and selects the corresponding state representation. The second network, φ, learns task-specific state representations. This gated network architecture is similar to the one proposed by Droniou et al. (2015). Their network

[*] The first two authors contributed equally to this work.

Workshop on Deep Learning for Action and Interaction at NIPS 2016, Barcelona, Spain.

simultaneously learns discrete classes jointly with continuous class variations (called *submanifolds*) in an unsupervised way. In our approach, we learn discrete tasks rather than discrete classes; we learn task-specific state representations rather than class-specific submanifolds. We train the two coupled networks simultaneously using the *robotic priors* learning objective (Jonschkowski & Brock, 2015), exploiting physics-based prior knowledge about how states, actions, and rewards relate to each other. Both networks learn from raw sensor data, without supervision and solely based on the robot's experiences.

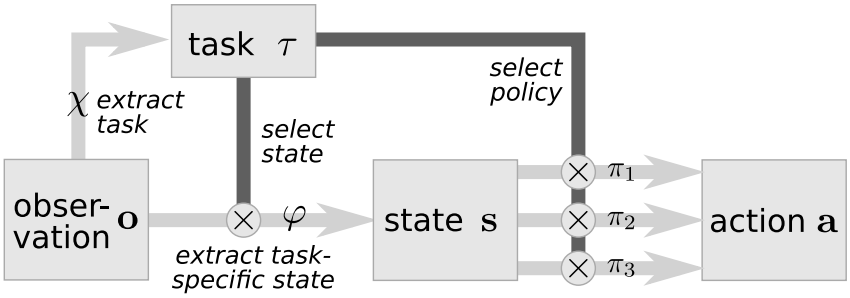


Figure 2: Overview of the gated network for state representation learning for multiple tasks.

We show in simulation experiments that MT-LRP is able to learn multiple state representations and task detectors from raw observations and that these representations allow to learn better policies from fewer data when compared with prior state representation learning methods. Moreover, we analyze the contribution to this result of each the method's individual components.

2 Multi-Task State Representation Learning: MT-LRP

We formulate MT-LRP in a reinforcement learning (RL) setting using a Markov decision process (MDP) (S, A, T, R, γ). We consider an episodic setting with episodes of finite length, a continuous state space S and a discrete action space A.

State Representation Learning for Reinforcement Learning We assume that the agent *cannot* directly observe the state $\mathbf{s}$ but only has access to observations $\mathbf{o} \in O$, which are usually high-dimensional and contain task-irrelevant distractors. This requires the agent to extract the state from observations by learning an observation-state-mapping $\varphi : O \to S$, and use the resulting state representation S to solve the RL problem. To learn the state representation, we apply *learning with robotic priors* (Jonschkowski & Brock (2015), from now on referred to as *LRP*). This method learns φ from a set of temporally ordered experiences $D = \{(\mathbf{o}_t, a_t, r_t)\}_{t=1}^{d}$ by optimizing an objective function $\mathcal{L}_{\mathrm{RP}}(D, \varphi)$ that expresses different priors about suitable state representations for robot RL. We optimize it using gradient descent, assuming φ to be differentiable. For more information regarding the robotic prior objective, we refer the reader to the supplementary material.

Multi-Task State Representations Consider a scenario in which an agent is learning multiple distinct tasks. For each task $\tau \in \{1, \ldots, T\}$, the agent now requires a task-specific policy $\pi_\tau : S_\tau \to A$. We approach the problem by learning a task-specific *state representation* $\varphi_\tau : O \to S_\tau$ for each policy, and a *task detector* χ which determines the task, given the current observation. We will consider a probabilistic task-detector $\chi : O \to [0,1]^T$ that assigns a probability to each task being active.

In order to solve the full multi-task RL problem, we must learn χ, $\{\varphi_\tau\}_{\tau \in \{1,\ldots,T\}}$ and $\{\pi_\tau\}_{\tau \in \{1,\ldots,T\}}$. We propose to address this problem by MT-LRP, a method that jointly learns χ and $\{\varphi_\tau\}_{\tau \in \{1,\ldots,T\}}$ from raw observations, actions, and rewards. MT-LRP then uses the state representations $\{\varphi_\tau\}$ to learn task-specific policies $\{\pi_\tau\}_{\tau \in \{1,\ldots,T\}}$ (using standard RL methods), and switches between them using the task detector χ. To solve the joint learning problem, MT-LRP generalizes LRP (Jonschkowski & Brock, 2015) in the following regards: (i) we replace the linear observation-state-mapping from the original method with a *gated neural network*, where the gates act as task detectors that switch between different task-specific observation-state-mappings; (ii) we extend the list of robotic priors by the prior of *task coherence*, which allows us to train multiple task-specific state representations without any specification (or labels) of tasks and states.

Gated Neural Network Architecture We use a gated neural network architecture as shown schematically in Fig. 2. The key idea is that both the task detector χ as well as the state representation φ are computed from raw inputs. However, the output of the task detector *gates* the output of the state representation. Effectively, this means the output of $\chi(\mathbf{o})$ decides which task-specific state representation φ_τ is passed further to the policy, which is also gated by the output of $\chi(\mathbf{o})$.

Formally, $\chi(\mathbf{o}) = \sigma(\chi_{\text{pre}}(\mathbf{o}))$ is composed of a function χ_{pre} with T-dimensional output and a *softmax* σ that ensures that χ computes a proper probability distribution over tasks. The probabilities are then used to gate φ. To do this, we decompose φ into a *pre-gating function* φ_{pre} that extracts features shared across all tasks (i.e. "multi-task" in the sense of Caruana (1997)), and a $T \times M \times N$ *gating tensor* $\mathbf{G}$ that encodes the T (linear) observation-state mappings ($M = \dim(\mathbf{s})$ and N is the output dimension of φ_{pre}). The value of the state's i-th dimension $\mathbf{s}_i$ computes as the expectation of the dot product of gating tensor and $\varphi_{\text{pre}}(\mathbf{o})$ over the task probabilities $\chi(\mathbf{o})$:

$$s_i = \varphi_i(\mathbf{o}) = \sum_{k=1}^{T} \chi_k(\mathbf{o}) \, \langle \mathbf{G}_{k,i,:}, \varphi_{\text{pre}}(\mathbf{o}) \rangle. \tag{1}$$

Learning Objective To train the network, we extend the robotic prior loss $\mathcal{L}_{\text{RP}}$ by a *task-coherence prior* $\mathcal{L}_\tau$:

$$\mathcal{L} = \mathcal{L}_{\text{RP}}(D, \varphi) + \omega_\tau \mathcal{L}_\tau(D, \chi), \tag{2}$$

where ω_τ is a scalar weight balancing the influence of the additional loss term. Task coherence is the assumption that a task only changes between training episodes, not within the same episode. It does not presuppose any knowledge about the number of tasks or the task presented in an episode, but it exploits the fact that task switching weakly correlates with training episodes. It applies directly to the output of the task detector, $\chi(\mathbf{o})$, and consists of two terms:

$$\begin{aligned} \mathcal{L}_\tau^{\text{con+sep}} &= \mathcal{L}_\tau^{\text{con}} + \mathcal{L}_\tau^{\text{sep}} \\ &= \mathbf{E}\Big[H(\chi(\mathbf{o}_{t_1}), \chi(\mathbf{o}_{t_2})) \;\Big|\; \text{episode}_{t_1} = \text{episode}_{t_2} \Big] + \mathbf{E}\Big[e^{-H(\chi(\mathbf{o}_{t_1}), \chi(\mathbf{o}_{t_2}))} \;\Big|\; \text{episode}_{t_1} \neq \text{episode}_{t_2} \Big], \end{aligned} \tag{3}$$

where H denotes the cross-entropy $H(p,q) = -\sum_x p(x) \log q(x)$. The first term $\mathcal{L}_\tau^{\text{con}}$ enforces *task consistency* during an episode. We use it to penalize χ if it assigns different task distributions to inputs $\mathbf{o}_{t_1}, \mathbf{o}_{t_2}$ that belong to the same episode. The second term $\mathcal{L}_\tau^{\text{sep}}$ expresses *task separation* and encourages χ to assign tasks to different episodes. This loss is complementary to task consistency, as it penalizes χ if it assigns similar task distributions to $\mathbf{o}_{t_1}, \mathbf{o}_{t_2}$ from different episodes.

3 Experiments

We evaluate MT-LRP in the *multi-task slot-car racing* scenario (inspired by Lange et al. (2012)), where the agent controls one of multiple cars (Figure 1), with the goal of traversing the circuit as fast as possible without leaving the track due to speeding in curves. However, the agent does not know a priori which car it controls, and only receives the raw visual signal as input. Additionally, uncontrolled cars driving at random velocity, act as visual distractors. We turn this scenario into a multi-task problem in which the agent must learn to control *each* car, where controlling the different cars corresponds to separate tasks. To indicate the car that the agent controls, we indicate a picture of the car in the upper left corner. We provide the technical details of our experimental set-up as well as additional experiments, including a second *mobile navigation* scenario, in the supplementary material.

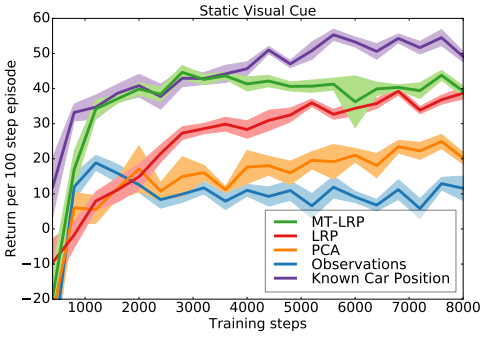


Figure 3: Reinforcement learning curves (mean and standard error) for different state representations for the *two*-slot car scenarios (static visual cue).

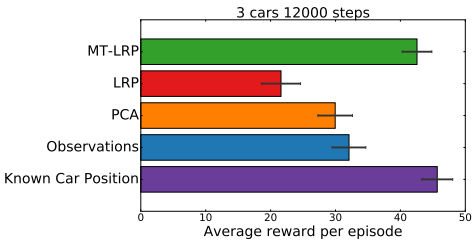


Figure 5: Reinforcement learning performance in the *three*-slot car scenario with static visual cue.

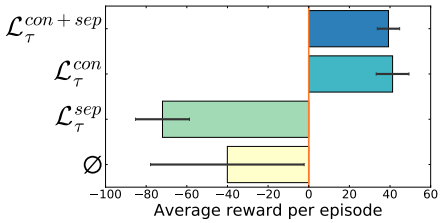


Figure 6: Task coherence: Average reward per episode (8000 samples).

3.1 Results

We will now present the main results of our experiments: (i) we show that MT-LRP enables the agent to extract better representations for RL; (ii) we provide insight in how the learner detects/learns task and state representations; (iii) we show the contribution of each of the task-coherence loss terms.

MT-LRP Extracts Better State Representations for RL Figure 3 shows the learning curves for RL based on state representations learned by the different methods in the two-slot-car scenario (*static visual cue* condition, see supplementary material). No method reaches the performance of the upper baseline, mainly due to aliasing errors resulting from the low image resolution. MT-LRP gets very close to the performance of the upper baseline, especially for very low amounts of training data ($d < 2500$), whereas LRP does not even attain this level of performance for the full training set $d = 8000$ as LRP can only learn to extract the position of all cars, not the relevant one. The gap between MT-LRP and LRP increases even more if we add another car (Figure 5).

MT-LRP Detects All Tasks and Learns Good State Representations To gain more insight into what is learned, we analyze the state representations extracted by MT-LRP in Figure 4. Each point in the figure corresponds to one observation, markers indicate the task and colors the most active gate unit. We see that the first gate unit (blue) is always active for task 1 (circle), and the second gate unit for task 2, and that the states reflect the circular structure of the slot car racing track. We thus conclude that MT-LRP has learned to identify the tasks and to represent the position of each car on the track.

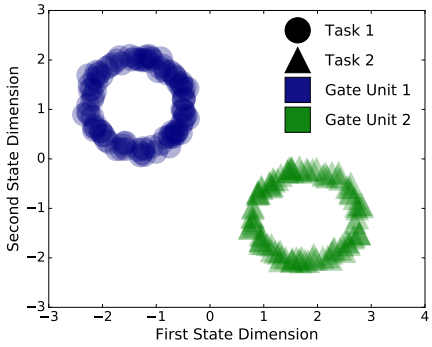


Figure 4: State representation learned per task (different markers) and per gate unit (different colors)

Task-Consistency is Critical for Learning Performance To understand the influence of the different task-coherence prior variants, we compared their performance in Figure 3.1. We see that relying solely on the robotic priors gives poor results, mainly because the gate units are not used properly: more than one gate unit is activated per task (χ has high entropy). Adding the task-separation prior forces the network to use as many gates as possible (5 in our case), leading to bad state representations. Interestingly, using task consistency only gives roughly the same result as using task consistency and task separation. This indicates that the robotics prior loss is sufficient to encourage the learner to separate different tasks: however, the task-consistency loss is required to guide the learned in identifying the tasks.

4 Conclusion

We have presented MT-LRP, a method for multi-task state representation learning with robotic priors. The method learns in an unsupervised fashion, solely based on the robots own observations, actions, and rewards. Our experiments confirmed that MT-LRP is effective at simultaneously identifying tasks and learning task-specific state representations. This capability is beneficial for scaling reinforcement learning to realistic scenarios that require dedicated skills for different tasks.

References

R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Alain Droniou, Serena Ivaldi, and Olivier Sigaud. Deep unsupervised network for multimodal perception, representation and classification. *Robotics and Autonomous Systems*, 71:83–98, September 2015. ISSN 0921-8890. doi: 10.1016/j.robot.2014.11.005.

Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, July 2015. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-015-9459-7.

Rico Jonschkowski, Sebastian Höfer, and Oliver Brock. Patterns for Learning with Side Information. *arXiv:1511.06429 [cs, stat]*, November 2015.

S. Lange, M. Riedmiller, and A. Voigtlander. Autonomous reinforcement learning on raw visual input data in a real world application. In *2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, June 2012. doi: 10.1109/IJCNN.2012.6252823.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *arXiv:1504.00702 [cs]*, April 2015.

José Martín H, Javier de Lope, and Darío Maravall. The knn-td reinforcement learning algorithm. *Methods and Models in Artificial and Natural Computation. A Homage to Professor Mira's Scientific Legacy*, pp. 305–314, 2009.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Justus Piater, Sébastien Jodogne, Renaud Detry, Dirk Kraft, Norbert Krüger, Oliver Kroemer, and Jan Peters. Learning visual representations for perception-action systems. *The International Journal of Robotics Research*, 30(3):294–307, March 2011. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364910382464.

Olivier Sigaud, Clément Masson, David Filliat, and Freek Stulp. Gated networks: an inventory. *CoRR*, abs/1512.03201, 2015. URL `http://arxiv.org/abs/1512.03201`.

Manuel Watter, Jost Tobias Springberg, Joschka Boedecker, and Martin Riedmiller. Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images. *arxiv*, 2015.

Laurenz Wiskott and Terrence J. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002.

Supplementary Material

Learning with Robotic Priors

We describe the *learning with robotic priors* (Jonschkowski & Brock (2015) used by MT-LRP. This method learns φ from a set of temporally ordered experiences $D = \{(\mathbf{o}_t, a_t, r_t)\}_{t=1}^{d}$ by optimizing the following loss:

$$\mathcal{L}_{\mathrm{RP}}(D,\varphi) = \omega_{\mathrm{t}}\mathcal{L}_{\mathrm{temp.}}(D,\varphi) + \omega_{\mathrm{p}}\mathcal{L}_{\mathrm{prop.}}(D,\varphi) + \omega_{\mathrm{c}}\mathcal{L}_{\mathrm{caus.}}(D,\varphi) + \omega_{\mathrm{r}}\mathcal{L}_{\mathrm{rep.}}(D,\varphi). \tag{4}$$

This loss consists of four terms, each expressing a different prior about suitable state representations for robot RL. We optimize it using gradient descent, assuming φ to be differentiable. We now explain the four robotic prior loss terms in Eq. (4).

Temporal Coherence enforces states to change gradually over time (Wiskott & Sejnowski, 2002):

$$\mathcal{L}_{\mathrm{temp.}}(D,\varphi) = \mathbf{E}\left[\|\Delta\mathbf{s}_t\|^2\right],$$

where $\Delta \mathbf{s}_t = \mathbf{s}_{t+1} - \mathbf{s}_t$ denotes the state change. (To increase readability we replace $\varphi(\mathbf{o})$ by $\mathbf{s}$.)
Proportionality expresses the prior that the same action should change the state by the same magnitude, irrespective of time and the location in the state space:

$$\mathcal{L}_{\text{prop.}}(D, \varphi) = \mathbf{E}\Big[(\|\Delta \mathbf{s}_{t_2}\| - \|\Delta \mathbf{s}_{t_1}\|)^2 \;\Big|\; a_{t_1} = a_{t_2} \Big].$$

Causality enforces two states $\mathbf{s}_{t_1}, \mathbf{s}_{t_2}$ to be dissimilar if executing the same action in $\mathbf{s}_{t_1}$ generates a different reward than in $\mathbf{s}_{t_2}$.

$$\mathcal{L}_{\text{caus.}}(D, \varphi) = \mathbf{E}\Big[e^{-\|\mathbf{s}_{t_2} - \mathbf{s}_{t_1}\|^2} \;\Big|\; a_{t_1} = a_{t_2}, r_{t_1+1} \neq r_{t_2+1} \Big].$$

Repeatability requires actions to have repeatable effects by enforcing that the same action produces a similar state change in similar states:

$$\mathcal{L}_{\text{rep.}}(D, \hat{\varphi}) = \mathbf{E}\Big[e^{-\|\mathbf{s}_{t_2} - \mathbf{s}_{t_1}\|^2} \|\Delta \mathbf{s}_{t_2} - \Delta \mathbf{s}_{t_1}\|^2 \;\Big|\; a_{t_1} = a_{t_2} \Big].$$

Additionally, the method enforces *simplicity* by requiring $\mathbf{s}$ to be low-dimensional.

Note that learning with robotic priors only makes use of the actions a, rewards r, and temporal information t during optimization, but not at test time for computing $\varphi(\mathbf{o}) = \mathbf{s}$. Using a, r and t in this way is an instance of the learning with side information paradigm (Jonschkowski et al., 2015).

Slot-Car Racing

Experimental Set-Up

The agent controls the velocity of one car (see Fig. 1), receives a reward proportional to the car's velocity, chosen from [0.01, 0.02, ..., 0.1], and a negative reward of -10 if the car goes too fast in curves. The velocity is subject to Gaussian noise (zero mean, standard deviation 10%) of the commanded velocity. All cars move on independent lanes and do not influence each other. The agent observes the scenario by getting a downscaled 16x16 RGB top-down view (dimension $N = 16 \times 16 \times 3 = 768$) of the car circuit (Fig. 1(b)).

In our experiments, there are two or three cars on the track, and the agent controls a different one in every episode. To recognize the task, the agent must be able to extract a visual cue from the observation which correlates with the task. We study two types of visual cues:
Static Visual Cue: The arrangement of cars stays the same in all episodes and a static visual cue (a picture of the controlled car) in the top-left image corner indicates which car is currently controlled.
Dynamic Visual Cue: The agent always controls the same car (with a certain color), but in each task the car is located on a different lane (as in Fig. 1(b)).

The results for the static-visual-cue experiment are presented in the main part of the paper, and for the dynamic-visual-cue experiment below.

Data Collection and Learning Procedure: The agent collects 40 episodes per task, each episode consisting of 100 steps. To select an action in each step, the agent performs ε-greedy exploration by picking a random action with probability $\varepsilon = 0.3$ and the best action according to its current policy otherwise. The agent computes a policy after every τ episodes, by first learning the observation-state mapping φ (state representation) and then computing policies $\pi_1, \ldots, \pi_\tau$ (based on the outcomes of the learned χ and φ). To monitor the agent's learning progress, we measure the average reward the agent attains on T test episodes, i.e. one test episode of length 100 per task (using the greedy policy), amounting to 8000 experiences in total. To collect sufficient statistics, the whole experiment is repeated 10 times.
Policy Learning: We consider the model-free setting with continuous states S, discrete actions A and solve it using nearest-neighbor Q-learning *kNN-TD-RL* (Martín H et al., 2009) with $k = 10$. More recent approaches to model-free RL would be equally applicable (Mnih et al., 2015).
Learning Strategies and Baselines: We compare five strategies. We run a) MT-LRP with 5 gate units (two/three more than necessary), state dimensionality $M = 2$ and using $\mathcal{L}_\tau^{\text{con+sep}}$ as task-coherence prior. We compare MT-LRP to several state representation methods; for each method we evaluate different M and report only the best performing M: a) robotic priors without gated network, LRP ($M = 4$), b) principal components analysis (*PCA*) on the observations ($M = 20$) and c) raw

observations ($M = 768$). Additionally, we evaluate d) a lower baseline in the form of a randomly moving agent and e) an upper baseline by applying RL on the known 2D-position of the slot car under control ($M = 2$). In Figure 3, the random baseline was omitted as it ranges around an average reward of -84.9 with standard error 0.72.

We use the same RL algorithm for all methods. To learn the state representations with robotic priors, we base our implementation on Theano and lasagne, using the Adam optimizer with learning rate 0.005, batch size 100, Glorot's weight initialization and $\omega_t = 1, \omega_p = 5, \omega_c = 1, \omega_r = 5, \omega_\tau = 10$. Moreover, we apply an L1 regularization of 0.001 on φ.
Additionally, we analyze the contribution of task coherence priors by applying MT-LRP to the full set of 8000 experiences a) without task-coherence, b) with task consistency $\mathcal{L}_\tau^{\text{con}}$ only c) with task separation $\mathcal{L}_\tau^{\text{con}}$ only) and d) without task consistency and separation $\mathcal{L}_\tau^{\text{con+sep}}$.

Results: Dynamic Cue Scenario

The results for the dynamic-visual-cue scenario are shown in Figure 7. We see that LRP-4 (i.e. with state dimensionality 4) performs on par with MT-LRP. We hypothesize that, for the dynamic cue, LRP is able to extract the position of the car on regardless of which lane it is in using a single linear mapping. Figure 8 confirms this hypothesis: LRP filters for the car's color (blue) along the track and assigns increasing weights to these pixels which results in the extraction of its position. It also assigns constant weights along the track in the red channel using the lane change of the two cars as an offset. This results in a mapping to two circles similar to Fig. 4, where the state encodes both the position and the task. Such a mapping can be expressed by a linear function precisely because the features that are relevant for one task do not reappear in another task (e.g. a blue slot car in track 1 does not appear in the task where the blue car is in track 2). However, there exists no equivalent linear mapping for the static-visual-cue variant of the slot-car problem, because cars that are relevant for one task are also present in every other task.

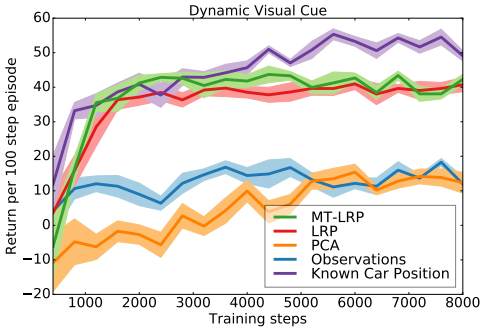


Figure 7: Reinforcement learning curves (mean and standard error) for different state representations for the *two*-slot car scenarios (dynamics visual cue).

We can generalize from this insight as follows. A single linear observation-state-mapping is sufficient for multiple tasks if the state representation for every task can be extracted by a linear function using only features that stay constant for all other tasks. If this is the case, than there is no need for decoupling the extraction of task and state.

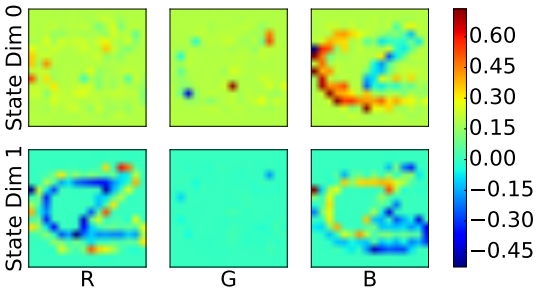


Figure 8: φ learned by LRP ($M = 2$) for the two-car dynamic visual cue tasks. Row corresponds to state dimension, column to RGB color channel.

Mobile Navigation

In the *mobile navigation* scenario (Figure 9), we study how MT-LRP scales to nonlinear problems. The agent has to navigate to the top-right corner of a room, only receiving partial views as observations. To enable the agent to localize, MNIST digits and a background pattern are printed on the ground, similar to the task proposed by Piater et al. (2011). Different arrangements of numbers and background patterns then correspond to different tasks.

Experimental Set-Up

The agent moves in a 60x60 pixel room by shifting up, down, left or right by 2 pixels. Motion is restricted to a range of 50x50 pixels. On the ground of the room, we arrange 9 random MNIST numbers, cropped to 20x20 pixels, in a 3x3 grid. As an observation, the agent receives a 20x20 image

centered at the agent's current location. The agent receives a positive reward of 100 when located in the upper right region ($x >= 30$ and $y >= 30$) of the room, and a negative reward of 1 when trying to move outside the 50x50 pixel range. We apply the agent to four tasks, each characterized by a different arrangement of numbers. Additionally, each arrangement is combined with a different background image (white, grey, horizontal and vertical stripes) in order to enable the task detector to extract the task from a single partial view.

Data Collection and Learning Procedure: The agent collects 10 episodes per task, each episode consisting of 50 steps. At the beginning of every episode, the agent is positioned at a random location and then moves randomly. In this scenario, we omit the RL part, and only assess the learned state representations visually, by comparing them to the ground truth position of the agent; our previous work showed that a representation sufficiently close to the ground truth position enables successful RL in such navigation scenarios (Jonschkowski & Brock, 2015).

Learning Strategies and Baselines: We apply MT-LRP to a network with 4 gate units and set the hyperparameters to $\omega_t = 1, \omega_p = 5, \omega_c = 2, \omega_r = 5, \omega_\tau^{con} = 10, \omega_\tau^{sep} = 50$.

Neural Network Structure: We use two convolutional neural networks of similar structure for χ_{pre} and φ_{pre}, each consisting of a pair of convolutional/max-pooling layers (16 filters of size 5x5, pool size 2x2) and a 32-dimensional fully connected layer mapping to the two-dimensional state representation. For the convolutional and fully connected layers, we use reLU activations.

Results

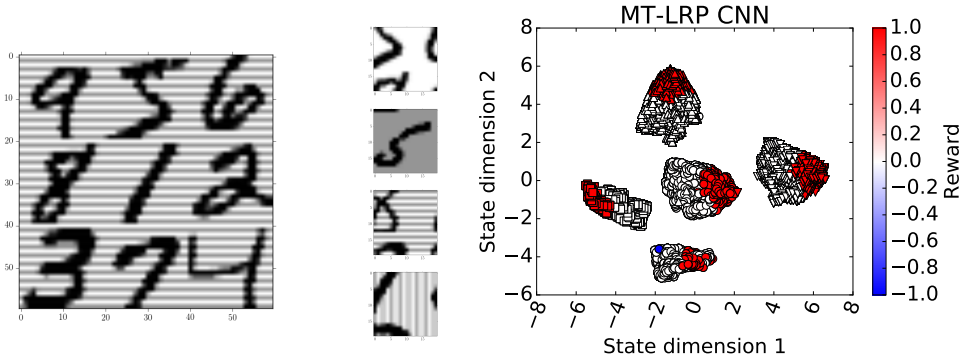


Figure 9: Mobile navigation scenario. Left: example room. Middle: example observations. Right: State representations extracted by MT-LRP. Color encodes reward, form the activated gate.

Figure 9 shows the representations extracted in the mobile navigation task. We see that MT-LRP identifies all tasks, but splits one task across two gates (resulting in five clusters and a task-detection accuracy of 0.899). All except one representation approximate the square room structure with the high-reward area in the upper right corner. However, dropping the task separation loss changes the result: MT-LRP then uses fewer gates and tends to map representations for *all* tasks to a single representation – LRP produces a similar result. The reason is the sufficiently high representational power of the used convolutional network architecture. Once the task gets more complex the gated structure is required, as shown in the slot-car experiments.