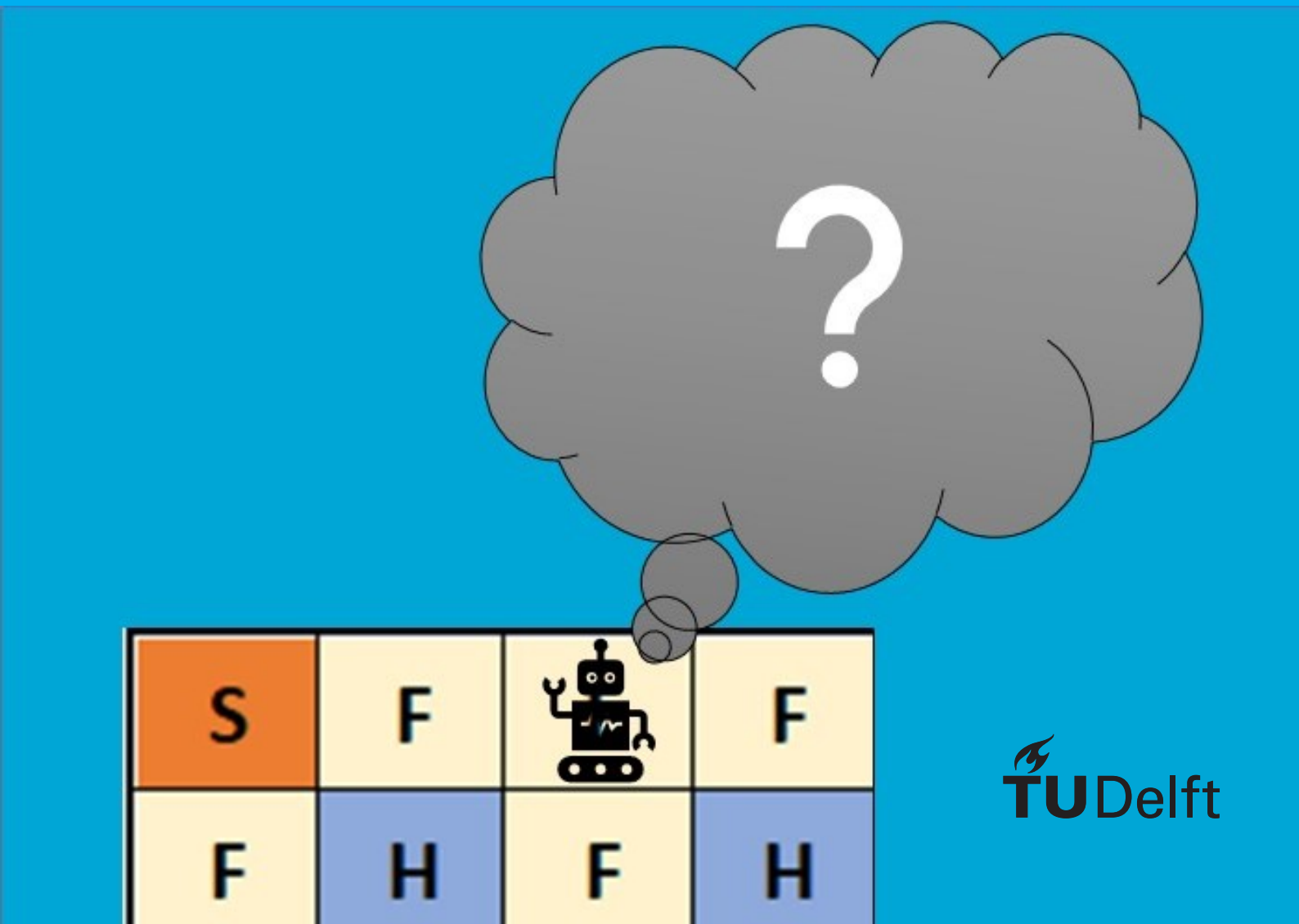
Learning What to Attend to

Using bisimulation metrics to explore and improve upon what a deep reinforcement learning agent learns



TU Delft

Learning What to Attend to

Using bisimulation metrics to explore and improve upon what a deep reinforcement learning agent learns

by

Nele Albers

to obtain the degree of

Master of Science Computer Science
with a specialization in **Data Science & Technology**

at the Delft University of Technology,
to be defended publicly on Wednesday August 12, 2020 at 2:00 PM.

Student number:	4853261	
Project duration:	December 18, 2019 – August 12, 2020	
Thesis committee:	Dr. F. Oliehoek,	TU Delft, supervisor
	Dr. M. T. J. Spaan,	TU Delft
	Dr. W. Brinkman,	TU Delft
	M. Suau de Castro,	TU Delft, daily supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis is the product of my graduation project as part of the Interactive Intelligence group at the Delft University of Technology. It is now almost 18 months ago that my supervisor Dr. Frans Oliehoek asked me whether I would be interested in writing my thesis on reinforcement learning. Knowing just the basics of reinforcement learning at that point, it is largely due to his enthusiasm and constant supply of interesting papers that I decided to give it a try, and I am glad that I did. During my studies, I have often felt frustrated by applying hyperparameters, algorithms or network architectures without precisely knowing *why* they work. Exploring what it is that deep reinforcement learning agents learn and using the gained insights to improve upon their learning thus has been a very enjoyable and satisfying experience for me. I hope that my results turn out to be useful for further advances in this direction.

This work would not have been possible without the advice and support of several people. I am thankful to my supervisor Dr. Frans Oliehoek for introducing me to the area of reinforcement learning, and to him and my other committee members Dr. Matthijs Spaan, Dr. Willem-Paul Brinkman and Miguel Suau de Castro for guiding this project. Thank you for making me enjoy the past nine months so much that I want to continue working in research. Especially Miguel I would like to also thank for his assistance in trimming a 250-page report draft. A special thanks to my boyfriend Anurag for helping me get things into perspective and reminding me of the value of breaks, albeit I am no longer allowed to use the term ”bisimulation metrics” in his presence, and to my friends for sharing their thesis experience and listening to mine. Last but not least, I would like to thank my family for giving me the right amounts of support, understanding, space and distraction at the right times.

Nele Albers
Delft, August 2020

Abstract

We analyze the internal representations that deep Reinforcement Learning (RL) agents form of their environments and whether these representations correspond to what such agents should ideally learn. The purpose of this comparison is both a better understanding of why certain algorithms or network architectures perform better than others and the development of methods that specifically target discrepancies between what is and what should be learned. The concept of ideal representation we utilize is based on stochastic bisimulation and bisimulation metrics, which are measures of whether and to which degree states are behaviorally similar, respectively. Learning an internal representation in which states are equivalent if and only if they are bisimilar and in which distances between non-equivalent states are proportional to how behaviorally similar the states are has several desirable theoretical properties. Yet, we show empirically that the extent to which such a representation is learned in practice depends on several factors and that a precise such representation is not created in any case. We further provide experimental results that suggest that learning a representation that is close to this target internal state representation *during* training may improve upon the learning speed and consistency, and doing so *by the end of* training upon generalization.

Contents

Abbreviations

CNN Convolutional Neural Network. 11

DQN Deep Q-network. 11, 27–33, 35–38, 42, 44, 45, 48, 49, 52–54, 58, 60, 61, 63–65, 73, 74, 79, 81, 84–87, 90, 92, 97, 109, 129, 133, 136–138, 146

DRQN Deep Recurrent Q-network. 11, 64, 121, 123, 125, 126

KL Kullback-Leibler. 19, 20

LSTM Long Short-Term Memory. 11, 12, 121, 124–126

MDP Markov Decision Process. 8–10, 12–14, 17, 24, 25, 32, 119

MSE Mean Squared Error. 47, 48

PCA Principal Component Analysis. 19

POMDP Partially Observable Markov Decision Process. 9–11, 17, 119

ReLU Rectified Linear Unit . 121, 146

RL Reinforcement Learning. 1–7, 10, 12, 13, 19, 21, 22, 27, 45, 46, 63–65, 119

RNN Recurrent Neural Network. 11

SGD Stochastic Gradient Descent. 11

t-SNE t-Distributed Stochastic Neighbor Embedding. 19, 20, 22, 23, 29, 31, 32, 35, 37, 39, 40, 42, 64, 72–74, 84, 92, 119, 121–126, 142, 148

List of Symbols

c_K Pearson correlation coefficient between the maximum Kantorovich distances T_K of the transition functions of states and the Euclidean distances between the activations states are mapped to in a network layer

$c_{d'_{fix}}$ Pearson correlation coefficient between the distances of states with respect to d_{fix} as approximated by means of the algorithm by [8] and the Euclidean distances between the activations states are mapped to in a network layer

$c_{K(d_{fix})}$ Pearson correlation coefficient between the distances of states based on d_{fix} and the Euclidean distances between the activations states are mapped to in a network layer

c_{Q^*} Pearson correlation coefficient between the maximum absolute distances of states with respect to Q^* and the Euclidean distances between the activations states are mapped to in a network layer

c_{Rew} Pearson correlation coefficient between the maximum absolute reward distances of states and the Euclidean distances between the activations states are mapped to in a network layer

c_{TV} Pearson correlation coefficient between the maximum total-variation distances T_{TV} of the transition functions of states and the Euclidean distances between the activations states are mapped to in a network layer

T_K Kantorovich distance between the transition functions of states

T_{TV} Total-variation distance between the transition functions of states

List of Tables

1 Introduction

Recent years have seen a surge of algorithms and architectures for deep *Reinforcement Learning* (RL), many of which have shown remarkable success for various problems. Yet, little work has attempted to relate the performance of these algorithms and architectures to what the resulting deep RL agents actually learn, and whether this corresponds to what they should ideally learn. Such a comparison may allow for both an improved understanding of why certain algorithms or network architectures perform better than others and the development of methods that specifically address discrepancies between what is and what should be learned. This thesis thus explores experimentally the internal state representations a deep RL agent creates of its environment to see whether these are in line with our theoretical expectations. Moreover, we empirically validate the usefulness of our concept of ideal internal state representation with regards to learning speed and generalization and develop a method that allows deep RL agents to learn closer to what they should.

1.1. Learning What to Attend to

When we speak of what a deep RL agent learns, we mean the internal representation that a neural network forms of the environment. That is, the activation patterns that arise in each neural network layer as the result of feeding observations or observation histories to the network. Thereby, if observations with different values for a feature are mapped to the same activation, the agent does not attend to the feature and the feature hence cannot inform the agent's action choice. For example, a firefighter robot may map the observations ”smoke above blue house” and ”smoke above orange house” to the same activation, thus ignoring the colors of the houses. Similarly, if the agent learns to pay attention to a past observation, it distinguishes histories of interactions with the environment that differ in whether or not they contain this observation. For instance, a firefighter robot could disregard whether or not it has encountered a stray dog before seeing a burning house.

Why does a deep RL agent need to learn what to attend to?

The perceptions or observations the agent receives from its environment through sensors can be limited in two non-exclusive ways:

- In the first scenario, the *observations contain redundant or superfluous information* that is not necessary to learn to act optimally in the environment. For example, a firefighter robot does not have to consider the colors of the cars in front of a house when deciding whether to extinguish a fire, yet its camera may supply this information. In such a case, an agent with limited computational and memory resources needs to disregard the colors of cars and pay selective attention to solely relevant features [42].
- In the second scenario, *each single observation does not contain enough information* to learn to act successfully based on this observation alone. A firefighter robot that looks at the wall of a house, for instance, does not know based on seeing this wall whether or not there is a fire in this house. In such a case, an agent suffers from the hidden state problem, which can often be solved by keeping a memory of previous interactions with the environment [42]. For example,

the firefighter robot may still have the previous observation of smoke above the house's roof in memory.

While these two scenarios appear very different at first glance, they actually are closely related [42]. In both cases, an agent with limited computational and memory resources needs to learn which features of its previous and current perceptions to pay attention to. When paying attention to a feature, an agent distinguishes observations with different values for this feature, and the combination of all such distinctions an agent makes forms its internal state space or internal state representation. This internal state representation may consist of multiple internal states, and each observation or history of observations from the environment is mapped to exactly one internal state. As multiple (histories of) observations can be mapped to the same internal state if they differ in solely a superfluous feature value or irrelevant past observation, the internal state representation is an abstraction of the state space of the environment and each state in the internal state representation is an abstract state.

What should a deep RL agent ideally learn to attend to?

There are several criteria that the internal state representation formed by a deep RL agent should ideally meet:

- The internal state representation should allow the agent to learn to act optimally in a domain. If the internal state representation enables the agent to predict both the next reward and the next state for each action based on the internal state that an observation (history) is mapped to, the internal state representation is said to be *Markov* and the agent is guaranteed to find the optimal action for each observation (history).

- To minimize the amount of data, time and memory required for training, the internal state representation should have as few internal states as possible during training. More precisely, rather than allowing the prediction of *all* features of next states, the state representation solely has to enable the prediction of *relevant* features of next states. Relevant features thereby are those that are necessary to in turn predict the reward and relevant features of next states. For example, if the colors of houses are not important, the colors should be ignored. Hence, observations of houses differing solely in a house's color should be mapped to the same activation pattern. This is especially important for domains with high-dimensional observations such as images.

- The created internal state representation should allow the agent to quickly adapt to rewards or transition probabilities slightly different from those experienced during training, as long as the modifications to the transition and reward functions do not cause formerly irrelevant features to be relevant. In robotics problems, for instance, domain shifts may arise [29].

- Since transition and reward functions are commonly estimated, we would like the formed internal state representation to be relatively insensitive to minor approximation errors. Consequently, we would like the Euclidean distances between internal states to correspond to *how* "behaviorally different" [11] the (histories of) observations mapped to those internal states are.

The criterion of allowing for the prediction of the reward and of relevant features of next states is met by the *coarsest Markov state representation*. It maps (histories of) observations to the same internal state if and only if they have the same expected reward and the same transition distribution over all other internal states for all actions. Thus, it is the smallest state space that still allows for the prediction of the next reward and the next state [21]. Finally, as long as modifications to the transition and reward functions do not cause states with the same rewards and transition distribution over all other internal states on the original domain to behave differently, the coarsest Markov state representation for the original domain is sufficient to learn to act optimally in such a modified domain.

Yet, the coarsest Markov state representation by itself does not fulfill the criterion that Euclidean distances between internal states should be proportional to *how* behaviorally different the (histories of) observations mapped to those internal states are. This is the case, because the notion of coarsest Markov state representation stems from the context of state abstraction and thus does not impose any specific distances between internal states. To obtain such Euclidean distances, we make use of *bisimulation metrics* [12], which are indeed measures of how behaviorally similar states are. More precisely, we want Euclidean distances to correspond to the distances assigned by a specific bisimulation metric,

which gives a distance of zero to states if and only if they are mapped to the same internal state in the coarsest Markov state representation. For ease of notation, we will refer to the resulting state representation simply as the coarsest Markov state representation throughout our work. Yet, it is important to keep in mind that this state representation has specific Euclidean distances between internal states, unlike the coarsest Markov state representation as it is defined in the context of state abstraction.

Why do we want to know what a deep RL agent learns?

In recent years, state-of-the-art RL approaches in various research areas such as drug design [49], autonomous helicopter flight [45] and financial trading systems [6] commonly make use of neural networks. The main reasons for the popularity of neural networks are that they learn features directly from data and allow for end-to-end learning, thus rendering feature engineering unnecessary and reducing the need for domain knowledge [37]. Especially, the fact that neural networks can extract effective low-dimensional features from high-dimensional data such as images and speech has enabled the scaling of RL algorithms to problems with large state or action spaces [3]. However, the end-to-end character and flexibility of neural networks often make it difficult to understand what exactly a neural network is learning.

As a consequence, designing a network architecture and training hyperparameters tends to resemble ”alchemy” rather than science [37], and even successful RL agents sometimes overfit to the training data instead of learning what we would like them to learn. Regarding the former, some approaches that learn neural network architectures [66] or quickly test candidate architectures with random weights [56] exist in the context of classification and could potentially be extended to RL, but such methods do not provide thorough insights into why one architecture may be more effective than another. Thus, the hope is that by understanding better what deep RL agents learn, we can make a more informed decision on how to design and train them.

1.2. Research Objective

Our research is structured along two primary questions:

1. Which internal state representations do deep RL agents form during training and how similar are these to the coarsest Markov state representation?

2. To which degree is creating internal state representations that are similar to the coarsest Markov state representation useful in practice?

To answer the first research question, we look at the internal state representations learned by deep RL agents at various stages during training and under different training conditions, and compare them to the coarsest Markov state representation. Moreover, to elucidate the usefulness of learning internal state representations that are similar to the coarsest Markov state representation in practice for the second research question, we strive to answer the following two sub-questions:

2a. To which degree does creating internal state representations that are more similar to the coarsest Markov state representation *during* training improve upon the learning speed and consistency of deep RL agents?

2b. To which extent does learning internal state representations that are more similar to the coarsest Markov state representation *by the end of* training lead to improved generalization?

To respond to these two sub-questions, we compare the learning speeds and consistencies and the generalization performances of neural networks with hidden-layer representations that differ in how similar to the coarsest Markov state representation they are, while controlling for other factors.

1.3. Related Work

Exploring what deep RL agents learn. The primary objective of our research is to contribute to interpreting what and how deep RL agents learn. Related work in this regard is the one by [30], which argues that non-stationarity may hurt generalization, especially when it occurs at the beginning of training. Yet, the authors do not specifically explore which internal state representations are formed at various stages during training to possibly explain these observations. To determine what an agent has learned, we

propose using several measures based on bisimulation metrics that denote how Markov an internal state representation is with respect to different model components. Other research has suggested to employ saliency maps [23] or t-SNE plots [44, 64] to visualize what an agent has learned, the latter of which we also utilize as supporting evidence. These approaches result in figures that are easy to understand, but they do not produce measures to effectively summarize the characteristics of an internal state representation. Thus, to compare state representations present at different points during training, one has to look at multiple images and deduce based on domain knowledge what an agent has learned. Moreover, while plotting predicted state-action values for certain states during training as in the work of [44] allows to easily track the development throughout training, doing so does not incorporate knowledge about the entire internal state representation. Lastly, to the best of our knowledge, no other work has previously computed how similar to the coarsest Markov state representation an internal state representation is to gain insights into the learning process.

State abstraction. Our computation of behavioral similarity is based on the notion of stochastic bisimulation [21] and bisimulation metrics [12], which were originally introduced in the context of state abstraction for fully observable environments. Stochastic bisimulation regards states as equivalent or bisimilar if and only if they have the same reward and the same transition distribution over all other state equivalence classes for all actions. Bisimulation metrics, on the other hand, are perceivable as a quantitative version of stochastic bisimulation in that they assign a distance of zero only to bisimilar states and that if the parameters of two bisimilar states are altered on a small scale, the metric distance between the two states will stay small.

Representation learning based on bisimulation metrics. We design an auxiliary loss based on bisimulation metrics to learn more useful internal state representations. In this context, the work most closely related to ours is the concurrent research by [64], which also proposes learning internal state representations based on bisimulation metrics. Yet, while the authors of [64] suggest to create an internal state representation in which distances between states correspond to how behaviorally different they are *under an optimal policy*, we propose to take *all actions* into consideration. Thus, a representation that is learned by means of the approach of [64] is no finer than one that is formed when using our auxiliary loss. The former therefore is sufficient to learn an optimal policy for only a subset of the changes made to the reward and the transition function that still allow for generalization based on the representation that we propose to learn. Moreover, [64] calculate behavioral similarity based on the latent space by assuming deterministic rewards and Gaussian transitions. As we compute behavioral similarity based on the true reward and transition functions, we make no such assumptions. Yet, the approach by [64] is more scalable to large domains, both because the computation time is lower and because the resulting state representation is no larger than the one formed by means of our method. Another related work is the one by [20], which also introduces an auxiliary loss based on bisimulation metrics. However, the Euclidean distances between states in the state representations learned by means of the auxiliary loss of [20] provide an *upper bound* to bisimulation metric-based distances. In our proposed state representation, on the other hand, the Euclidean distances are *exactly proportional* to the distances assigned by bisimulation metrics. Lastly, bisimulation metrics are costly to exactly compute in practice, as they require an enumeration of the state space and knowledge of the reward and transition functions. Therefore, the authors of [59] propose to employ the more general notion of *MDP homomorphism metrics* for representation learning. MDP homomorphism metrics differ from bisimulation metrics in that actions are also abstracted such that the effects of actions are maintained.

Representation learning based on other notions. The auxiliary loss we design introduces a bias to the learning process. Several other approaches to bias the representation learning of deep RL agents have been proposed in the literature. For example, the works of [31] and [17] put forward auxiliary losses based on predicting the next reward or the discount factor. Such methods tend to be successful in practice, but do not have strong theoretical foundations. Furthermore, rather than biasing the learning of deep neural networks by means of auxiliary losses, other work has proposed different models to learn more useful representations such as by incorporating ideas from symbolic reasoning [19]. For instance, the work of [55] constrains neural networks to capture typical characteristics of relational reasoning, which is the reasoning about the relations between objects and their characteristics. Another approach to learning more useful representations is to specifically focus on factors that may hurt generalization.

For example, the research by [30] improves generalization by reducing the non-stationarity an agent encounters during training. Moreover, the authors of [29] adapt several regularization techniques from the context of classification that are based on injecting noise during training to RL.

Transfer learning. Part of our work empirically investigates whether transferring representations that are more similar to the coarsest Markov state representation from a source domain to a related domain, which has different reward or transition functions but the same state-action space and a subset of the relevant features[1] of the original domain, improves upon the learning speed and accuracy on the related domain. Thus, we are concerned with whether the specific knowledge gathered from one domain in form of the coarsest Markov state representation can be employed to bias the learning process on a new task, thereby reducing the amount of data and time required for learning the new task [35]. Methods to accomplish such a knowledge transfer differ with respect to the number of source tasks used to bias the learning on a target task, the differences between the source and target tasks, the type of knowledge that is transferred, and the goal of the knowledge transfer [35]. A comprehensive survey of approaches to transfer learning can be found in [35]. Yet, to the best of our knowledge, no previous research has empirically tested the usefulness of learning the coarsest Markov state representation on an original domain with regards to generalization to a related domain with different reward or transition functions but a subset of the relevant features of the original domain.

1.4. Contributions

We split our contributions into *methodological* and *experimental* ones. Our methodological contributions are as follows:

- We propose using correlation coefficients based on bisimulation metrics to measure how similar to the coarsest Markov state representation an internal state representation is. These correlation coefficients also allow to specifically determine whether an internal state representation is Markov with respect to the rewards or Markov with respect to the transitions[2] (Chapter 3).
- We introduce an auxiliary loss that pushes a neural network to learn an internal state representation that is similar to the coarsest Markov state representation in a network layer (Chapter 4).

We further provide experimental contributions:

- We identify three overlapping learning phases that together make up the learning process of deep RL agents using model-free Q-learning agents as example. Thereby, it is during the second learning phase that internal state representations become increasingly similar to the coarsest Markov state representation. We also point out several factors that impact this learning process (Chapter 3).
- We show that learning a hidden-layer representation that is more similar to the coarsest Markov state representation *during* training can speed up the learning process and cause good solutions to be found more reliably (Chapter 4).
- We demonstrate that learning a hidden-layer representation that is more similar to the coarsest Markov state representation *by the end of* training leads to improved generalization to new irrelevant feature values. Creating such a representation also enables better generalization to related domains with modified reward or transition functions, as long as the modifications do not render formerly irrelevant features relevant (Chapter 5).

[1] Recall that we define relevant features as those that are needed to predict the reward and relevant features of next states.

[2] A state representation that is *Markov with respect to the reward* is one in which knowledge of previous internal states does not lead to a more accurate prediction of the next reward [42]. The definition of *Markov with respect to the transition* proceeds analogously.

1.5. Outline

The remainder of this work is organized as follows. In Chapter 2, we outline important background information with respect to the algorithms and terminology that we utilize. Our main results and the corresponding methodologies are subsequently described in Chapters 3, 4 and 5. Thereby, Chapter 3 sheds light on what deep RL agents learn and how similar the internal state representations are to the coarsest Markov state representation, and Chapters 4 and 5 investigate the usefulness of learning hidden-layer representations that are similar to the coarsest Markov state representation with regards to learning speed and generalization, respectively. Finally, Chapter 6 provides a summary of our findings and directions for future work.

2

Background

This chapter provides background information on the primary concepts that are important for this work. These are the coarsest Markov state representation in Section 2.1, fully and partially observable Markov decision processes in Sections 2.2 and 2.3, deep reinforcement learning in Section 2.4, state abstraction in Section 2.5, stochastic bisimulation and bisimulation metrics in Sections 2.6 and 2.7, and the t-SNE algorithm in Section 2.8.

2.1. Learning the Coarsest Markov State Representation

We suppose and experimentally verify in Chapters 4 and 5 that a deep RL agent should ideally learn an internal state representation in a hidden layer that is similar to the *coarsest Markov state representation*. In the context of state abstraction, the term coarsest Markov state representation refers to the unique abstraction of the state space that considers states as equivalent if and only if they have the same reward and the same transition distribution over all other state equivalence classes for all actions [21]. Yet, we use this term to refer to a state representation in which *all* Euclidean distances between states correspond to *how* "behaviorally different" [11] those states are. Notice that the coarsest Markov state representation as it is described in the context of state abstraction does not impose specific distances between non-equivalent states. This means that only the Euclidean distances of states mapped to the same internal state correspond to how behaviorally different those states are.

More precisely, we would like the Euclidean distances between internal states to be proportional to the *bisimulation metric*-based distances assigned to states. Bisimulation metrics[1] are based on the notion of *stochastic bisimulation* [21], which considers states as equivalent if and only if they are mapped to the same abstract state in the unique coarsest Markov state representation[2]. Such states that are equivalent under the notion of stochastic bisimulation are called *bisimilar*. Bisimulation metrics can be regarded as a quantitative version of stochastic bisimulation in that they assign a distance of zero only to bisimilar states and that if the parameters of two bisimilar states are altered on a small scale, the metric distance between the two states will stay small. Thereby, a bisimulation metric that is based on the Kantorovich distance[3] considers states as equivalent *if and only if* they are bisimilar, and it is thus this specific bisimulation metric that we would like Euclidean distances between states to correspond to[4].

The theoretical advantages of learning a hidden-layer representation that is similar to the coarsest Markov state representation are as follows:

- The coarsest Markov state representation is the smallest state representation that still allows for the prediction of the reward and the next state [21]. If an agent can predict the next reward and the next state, it is guaranteed to find an optimal policy based on (histories of) observations. Notice that while representing an optimal policy may require solely an abstraction of the state space

[1] See Section 2.7 for more information on bisimulation metrics.
[2] See Section 2.6 for details on stochastic bisimulation.
[3] The Kantorovich distance is also called Wasserstein distance, Monge-Kantorovich distance, Kantorovich-Rubinstein distance or earth mover's distance.
[4] Section 2.7.1 contains further information on this bisimulation metric.

that is coarser than the coarsest Markov state representation, a representation that suffices to *represent* an optimal policy is not necessarily sufficient for *learning* an optimal policy [42][5].

- The coarsest Markov state representation does not distinguish states based on features that are irrelevant for predicting the next reward and internal state. Thus, a policy learned based on this representation generalizes to different values for such features.
- As long as a subset of the features required for predicting the reward and the next internal state for an original domain is sufficient for predicting the reward and the next internal state after modifying the reward or the transition function, the coarsest Markov state representation for the original domain suffices to learn the Q-values of a thus modified domain.
- Making the Euclidean distances between internal states proportional to how behaviorally different states are renders the formed representation less sensitive to small estimation errors if the transition or reward functions are approximated.

Chapters 4 and 5 empirically show that learning a hidden-layer representation that is similar to the coarsest Markov state representation may improve upon both the learning speed and consistency on a single domain and the generalization to new superfluous feature values and modified reward and transition functions. Yet, while learning the coarsest Markov state representation is useful for generalization purposes and has strong theoretical properties, the coarsest Markov state representation may still contain too many states to render learning feasible for very large domains. Moreover, it may make more distinctions of states than would be necessary to represent the optimal policy at test time. Therefore, if low memory requirements of the learned model are important and strong generalization abilities are not needed, learning a coarser abstraction of the state space than the coarsest Markov state representation may be beneficial.

2.2. Markov Decision Process

A *Markov Decision Process* (MDP) is a tuple $\langle S, A, P, R\rangle$ where S is a finite state space, A is a finite action space, $P : S \times A \to \Pi(S)$ is the transition function such that $P(s'|s,a) \in [0,1]$ is the probability of arriving in state s' after taking action a in state s, and $R : S \times A \to \mathbb{R}$ is the reward function such that $R(s,a)$ is the instant reward for taking action a in state s[6]. S and A are commonly referred to as describing the *model shape*, whereas P and R are *model parameters*. Together, the model shape and the model parameters define a full MDP model. Note that the Markov property holds for the transitions and the rewards, because the state and reward at time t only depend on the state and action at time $t-1$ and not on the history of previous states or actions.

MDPs are a common choice for solving sequential decision problems in fully observable, stochastic environments in which the Markov property holds for transitions and rewards and rewards are assumed to be additive [53]. A deterministic policy $\pi : S \to A$ is a mapping from states to actions, whereas a stochastic policy $\pi : S \to \Pi(A)$ generates a probability distribution over actions. The goal in *Markov Decision Problems* typically is to learn a (potentially stochastic) optimal policy π^* that maximizes the expected cumulative (discounted) reward for acting in a given environment. One distinguishes finite-horizon and discounted infinite-horizon models depending on whether the time for acting is known and fixed to be T or assumed to be infinite [33]. In the former case, one seeks a policy to maximize

$$E\left[\sum_{t}^{T} r_t\right]$$

and in the discounted infinite-horizon case, one aims at maximizing

$$E\left[\sum_{t}^{\infty} \gamma^t r_t\right], \tag{2.1}$$

[5] Refer to Section 2.5 for more information on different levels of abstracting the state space.

[6] Some works use a different notation for the reward function in which the reward may depend only on the current state or also on the state s' the agent arrives at [21]. Such different notations do not radically change the problem [53].

where $0 \leq \gamma < 1$ is a discount factor that determines how much rewards obtained in the distant future are valued compared to rewards collected in the near future. In the discounted infinite-horizon case, an optimal stationary policy always exists, which means that the optimal action to take in a state does not depend on the time t. In finite-horizon models, however, such a policy is not guaranteed to exist, as the optimal action to take in a state may depend on the remaining amount of time. In the sequel, we assume a discounted infinite-horizon MDP unless otherwise noted.

For each policy π, $V^{\pi}(s)$ is the state value function that denotes the expected cumulative reward (see Equation 2.1) that is obtained by following π from state s and $Q^{\pi}(s,a)$ is the state-action value function that describes the expected cumulative reward for taking action a in state s and executing π thereafter. The state value function and state-action value function of an optimal policy π^* are denoted by $V^*(s)$ and $Q^*(s,a)$, respectively, where $V^* = max_{\pi} V^{\pi}$ and $Q^* = max_{\pi} Q^{\pi}$ [36]. If the whole model is given or estimated from samples, dynamic programming algorithms such as value iteration or policy iteration can be employed to exactly compute an optimal policy for an MDP. Both value iteration and policy iteration can be executed in polynomial time for a fixed γ, but the number of iterations needed for value iteration may scale with $\frac{1}{1-\gamma} log\left(\frac{1}{1-\gamma}\right)$ [39]. Value iteration thereby has a complexity of $O(|A||S|^2)$ per iteration [39], and each iteration of policy iteration consists of a policy improvement and a value determination step, whereby the former can be performed in $O(|A||S|^2)$ and the latter in $O(|S|^3)$.

If no full model is given and only samples of the form $\langle s,a,r \rangle$[7] are available, a model-free algorithm such as Q-learning [62] or SARSA [52] may be used to determine a policy instead of estimating the model parameters and then applying model-based approaches[8]. In Q-learning, $Q^*(s,a)$ is approximated independently of the policy followed by means of this update equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma max_{a' \in A} Q(s',a') - Q(s,a)\big), \tag{2.2}$$

where $0 \leq \alpha < 1$ is the learning rate. Q-learning converges as long as all state-action pairs are always updated [57]. Note that model-free approaches such as Q-learning and SARSA can also be employed based on samples collected by sampling from a model.

2.3. Partially Observable Markov Decision Process

A *Partially Observable Markov Decision Process* (POMDP) is a tuple $\langle S, O, A, R, P^o, P^a \rangle$ where S is a finite set of states, O is a finite set of observations, A is a set of actions, $R : S \times A \rightarrow \mathbb{R}$ is the reward function such that $R(s,a)$ is the instant reward for taking action a in state s, $P^o : S \times A \rightarrow \Pi(O)$ is the observation function such that $P^o(o|s',a) \in [0,1]$ denotes the probability of observing observation o after taking action a and reaching state s', and $P^a : S \times A \rightarrow \Pi(s)$ is the transition function such that $P^a(s'|s,a) \in [0,1]$ is the probability of arriving in state s' after taking action a in state s. Unless otherwise noted we make the assumption of an infinite-horizon POMDP where future rewards are discounted geometrically with a discount factor of $0 \leq \gamma < 1$.

Evidently, a POMDP model has the same components as an MDP model, supplemented by the set of observations O and the observation function P^o. The reason is that, unlike in an MDP, the agent cannot directly observe the current state as the observations are not Markovian. Consequently, a POMDP can be seen as too coarse of an abstraction in that the hidden state has been abstracted away or the observations have been grouped into too coarse equivalence classes. Thus, while the typical goal still is to find an optimal policy π^* that maximizes the expected cumulative discounted sum of rewards, such an optimal policy of a POMDP may employ the entire history of interactions with the environment to select the next action [43], whereas the optimal policy of an MDP requires solely the current state.

One common approach for acting optimally in a POMDP is for the agent to maintain an internal belief state b, which is a probability distribution over the states such that $b(s)$ denotes the probability that the current state is state s. Once the agent makes observation o after executing action a, a new belief $b'(s')$ for some state $s' \in S$ can be computed from $b(s)$ if the full POMDP model is known [33]:

$$b'(s') = \frac{P^o(o|s',a) \sum_{s \in S} P^a(s'|s,a) b(s)}{\sum_{s' \in S} P^o(o|s',a) \sum_{s \in S} P^a(s'|s,a) b(s)}. \tag{2.3}$$

[7] Many algorithms work with $\langle s,a,r,s' \rangle$- or $\langle s,a,r,s',a' \rangle$-samples. Clearly, a sequence of $\langle s,a,r \rangle$-tuples can be formed to take either format.

[8] See [53] for details on model-free algorithms.

Since this belief state is a sufficient statistic for the agent's history of interactions with the environment and its initial belief state, a POMDP can be reduced to a continuous MDP in which the fully observable states are the internal belief states[9].

Both value iteration and policy iteration can theoretically be applied to this continuous belief-state MDP. Thereby, the optimal value function with finite horizon t of a POMDP is piecewise linear and convex and can be described as the maximum of k $|S|$-dimensional α-vectors, which are linear functions that represent the value of a specific policy over t time steps:

$$V^*(b) = max_{i=1,\dots,k}(b\alpha_i). \tag{2.4}$$

Based on the t-step value function, new vectors to represent the value function for $t+1$ time steps can be found and the value function of an infinite-horizon POMDP can be approximated with arbitrary precision by a finite set of α-vectors [61]. Value iteration can be used to compute V^*, however, value iteration algorithms often make use of pruning operations due to the large number of $(|A|^{|O|})^t$ α-vectors [24][10]. This means that dominated α-vectors, which are vectors that do not impact the value function, are deleted. Yet, even when including pruning, exact value iteration can solely be used to solve small POMDPs. Similarly, exact policy iteration to find the optimal policy of a POMDP is only applicable to problems with tens of states [24][11].

2.4. Deep Reinforcement Learning

Deep RL methods approximate one or multiple RL components, such as the state value function, the state-action value function, the policy, or the transition or reward functions, by means of deep neural networks [37]. This work uses deep Q-learning, which means that the Q-function is approximated via a deep neural network. Note that standard Q-learning assumes that the Q-function learned by the agent can be stored exactly in a table. However, for domains with large state spaces, this typically is impossible. For instance, if the problem at hand is to assemble a machine that consists of $1,000$ parts and to track whether each of these parts are available, the standard representation would consist of 2^{1000} states [21]. Thus, for such problems, methods other than tables are used to represent the Q-function. Since $Q^*(s,a)$ may not be representable by the chosen representation, employing such representations is called *function approximation*.

The most simple form of function approximation is linear function approximation. Linear function approximation approaches are those that abstract the space of all (value) functions $f \in \mathbb{R}^{|S|}$ to the space of functions that can be represented as a weighted linear combination of I basis functions:

$$\overline{f} = \sum_{i \in I} w_i \phi_i,$$

where w_i is the weight assigned to basis function ϕ_i [41]. In linear value function approximation for MDPs, a value function thus is approximately described by such a weighted linear combination of features. In the context of POMDPs, however, the α-vectors a value function is based on are linearly approximated, since a value function of a POMDP is piecewise linear and convex (see Equation 2.4).

Compared to linear function approximators, non-linear ones allow for more flexible forms of abstraction. A common non-linear approach for abstracting (value) functions is to use neural networks. One reason for the increasingly widespread use of neural networks is that they learn their parameters directly from the data without any need for feature engineering. In addition, they are so flexible that even a simple feed-forward neural network with at least one non-linear hidden layer can represent an arbitrarily accurate approximation of any function, provided that it contains a sufficient number of hidden units [22]. Nevertheless, small networks that are fast to train are desirable in practice, and the fact that data is limited means that not all parameters are necessarily perfectly estimated.

2.4.1. Network Architectures

This work makes use of two types of neural networks for approximately representing Q-functions, one for fully and one for partially observable environments. These two types are described subsequently.

[9] Note that the states in an MDP can be seen as belief states in which $b(s_i) = 1$ for exactly one state $s_i \in S$ and $b(s_j) = 0$ for all $s_{j \neq i} \in S$ [53].

[10] One example of a value iteration algorithm that uses pruning is the incremental pruning algorithm. See Section 2.1 in [24] for details.

[11] See Section 2.2 in [24] for details on policy iteration in POMDPs.

Deep Q-Network. The *Deep Q-network* (DQN) proposed by [44] requires sequences of the form $\langle s_t, a_t, r_t, s_{t+1} \rangle$, where s_t is a stack of m frames, to learn the parameters θ of a deep *Convolutional Neural Network* (CNN)[12] that represents an approximately optimal state-action value function $Q(s, a|\theta)$. Importantly, whereas the Atari environments the DQN-agent was originally tested on are partially observable given a single observation, they are considered fully observable given the stack of $m = 4$ past frames that are fed into the DQN-agent at each time step. The DQN-agent could thus be seen as operating in a partially observable environment and keeping a memory of the past m observations. However, the method assumes the state to be fully observable via the input.

Specifically, $\langle s_t, a_t, r_t, s_{t+1} \rangle$-tuples are stored in an experience replay memory at each time step t. Then, during each learning iteration, samples $s, a, r, s' \sim U(D)$ are drawn from the experience replay memory uniformly at random to perform a Q-learning update. Each iteration i of Q-learning thereby aims at minimizing the following loss via *Stochastic Gradient Descent* (SGD)[13]:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s' \sim U(D)} \left[\left(r + \gamma max_{a'} Q(s', a'|\theta_i^-) - Q(s, a|\theta_i) \right)^2 \right], \tag{2.5}$$

where θ_i^- are the parameters of a target network, which are only updated with the main Q-network's parameters every c steps. Using an experience replay memory and a target network serves to improve learning stability. This is important, because Q-learning with non-linear function approximators is not guaranteed to be stable due to correlations between subsequent observations as well as between the Q-values and the target values.

Experimental results by [44] on Atari games reveal that states generating similar observations as well as those that lead to comparable expected rewards obtain similar abstract representations based on the network's activations. Together with the fact that the DQN-agent learns good policies in several Atari games, this suggests that the DQN-agent can extract the most relevant information from the high-dimensional observations. Yet, no bounds regarding the quality of the policy or value function found via the DQN-agent exist.

Deep Recurrent Q-Network. Since the optimal policy of a POMDP may utilize the entire history of interactions with the environment to select the next action, neural network architectures for POMDPs commonly make use of a *Recurrent Neural Network* (RNN). RNNs are neural networks for processing sequential data which keep an internal state h_t, which summarizes task-relevant information from the past sequence of data up to time t [22]. The output of an RNN at time t then is a function of the new data at time t and the internal state at time $t-1$. There are several ways for representing and updating such an internal state and consequently there are several types of RNNs. For example, one common architecture is a *Long Short-Term Memory* (LSTM) model, which was designed to be able to remember sequence parts from a more distant past [22][14].

The *Deep Recurrent Q-network* (DRQN) approach by [25] is designed for POMDPs. It replaces the first fully connected layer of the DQN-agent by an LSTM layer to introduce a more flexible form of memory, which can theoretically remember arbitrarily long sequences of observations[15]. In the DRQN-agent, the function $Q(o_t, h_{t-1}|\theta)$ is thus approximated rather than the function $Q(s_t, a_t|\theta)$ as in the DQN-agent, where $h_{t-1} = LSTM(h_{t-2}, o_{t-1})$ is the output of the LSTM layer at the previous time step [65]. To this end, the DRQN-agent is trained via sequences of the form $\langle o_t, a_t, r_t, o_{t+1} \rangle$ that are sampled from an experience replay memory.

Experimental results by [25] on a flickering version of an Atari domain, in which each frame is obscured with a certain probability, indicate that the DRQN-agent is able to detect important components

[12] See [22] for details on CNNs.

[13] Rather than using the entire available dataset to compute the loss as in gradient descent, stochastic gradient descent utilizes solely small batches of training data to estimate the loss. Importantly, the batch size is kept fixed as the amount of training data increases. Since the computational cost per model update thus does not depend on the size of the training dataset and SGD tends to converge before all training samples have been sampled for large datasets, SGD is commonly used for large datasets. However, the number of updates required to obtain convergence typically increases for larger datasets. Besides using pure SGD, adaptive learning rate optimization algorithms such as the Adam algorithm [34] are also popular for training neural networks. Such algorithms can be seen as extensions to SGD that are intended to combat the variance resulting from estimating the loss via mini batches instead of the entire training dataset. Refer to Chapter 5.9 in [22] for details.

[14] For more detailed information on RNNs see [22].

[15] The original DQN-agent takes a stack of m frames as input, which can be seen as utilizing a memory of fixed length m.

such as the velocity of objects in the game. This suggests that the internal state of the LSTM effectively summarizes past experience. However, the authors do not find a systematic advantage of using a recurrent network over stacking the input frames as in the DQN-agent across several tested flickering Atari games. Specifically, when granting a DQN- and a DRQN-agent access to the same number of past observations, no agent generally outperforms the other. Nevertheless, the DRQN-agent is found to generalize better to flickering versions of Atari games when trained on fully observable versions than the DQN-agent. This indicates that a recurrent layer may offer some robustness to missing information. Finally, this approach also does not provide theoretical guarantees regarding the quality of the learned approximate state-action value function.

2.5. State Abstraction

An RL agent with limited capacity needs to abstract the ground state space when forming an internal state representation. Formally, state abstraction methods group ground states into equivalence classes so that the value function takes on a constant value on all states within the same equivalence class [41]. To render solving problems with large state spaces computationally tractable, several approaches to abstract a given state representation have been proposed in the literature. Such abstraction approaches differ in how much information of the original model the resulting abstract representation maintains. Ideally, one wants to obtain the unique coarsest representation that is still Markovian [21], which is the model with the minimal Markovian state space. Yet, it may not be possible to find this minimal model efficiently. In the context of factored MDPs, for instance, [21] show that given a number n that is represented in unary notation, deciding whether the minimal model of a factored MDP has exactly n states if one knows that it has n or fewer states is NP-hard. Moreover, the coarsest Markovian representation may still be too large to allow for efficient planning, thus necessitating an even coarser abstraction.

In the area of MDPs, [36] distinguish state abstraction techniques depending on which of the components of the original MDP are maintained in the abstract representation. Letting $\overline{M} = \langle \overline{S}, A, \overline{P}, \overline{R} \rangle$ denote the abstract MDP that results from the original MDP M after applying the abstraction function $\phi : S \to \overline{S}$, $\phi(s) \in \overline{S}$ the abstract state the ground state s is mapped to, and $\phi^{-1}(\overline{s})$ for $\overline{s} \in \overline{S}$ the set of ground states that are grouped in the abstract state $\overline{s}$, the following five levels of abstractions are identified by [36]:

1. In a *model-irrelevance* abstraction ϕ_{model}, all states that are grouped in the same equivalence class have the same reward and transition functions so that the one-step model is maintained. Thus, if $\phi_{model}(s_1) = \phi_{model}(s_2)$, $R(s_1, a) = R(s_2, a)\ \forall a \in A$ and $\sum_{s' \in \phi^{-1}_{model}(\overline{s})} P(s'|s_1, a) = \sum_{s' \in \phi^{-1}_{model}(\overline{s})} P(s'|s_2, a)\ \forall a \in A, \forall \overline{s} \in \overline{S}$.
2. In a *Q^π-irrelevance* abstraction ϕ_{Q^π}, the state-action value function is preserved for all policies. Hence, if $\phi_{Q^\pi}(s_1) = \phi_{Q^\pi}(s_2)$, then $Q^\pi(s_1, a) = Q^\pi(s_2, a)\ \forall a \in A$.
3. In a *Q^*-irrelevance* abstraction ϕ_{Q^*}, the optimal state-action value function from the ground model is maintained. Therefore, if $\phi_{Q^*}(s_1) = \phi_{Q^*}(s_2)$, then $Q^*(s_1, a) = Q^*(s_2, a)\ \forall a \in A$.
4. In an *a^*-irrelevance* abstraction ϕ_{a^*}, the optimal action and its value are preserved. Thus, if $\phi_{a^*}(s_1) = \phi_{a^*}(s_2)$, then $Q^*(s_1, a^*) = max_a Q^*(s_1, a) = max_a Q^*(s_2, a) = Q^*(s_2, a^*)$, where $a^* \in A$ is the optimal action in the equivalence class of s_1 and s_2.
5. In a *π^*-irrelevance* abstraction ϕ_{π^*}, only the optimal action is preserved. That is, if $\phi_{\pi^*}(s_1) = \phi_{\pi^*}(s_2)$, then $Q^*(s_1, a^*) = max_a Q^*(s_1, a)$ and $max_a Q^*(s_2, a) = Q^*(s_2, a^*)$, where $a^* \in A$ is the optimal action in the equivalence class of s_1 and s_2.

Regarding the relationship between these levels of abstraction, [36] show that for any MDP it holds that $\phi_0 \succeq \phi_{model} \succeq \phi_{Q^\pi} \succeq \phi_{Q^*} \succeq \phi_{a^*} \succeq \phi_{\pi^*}$, where ϕ_0 is the ground representation with $\overline{S} = S$. This partial ordering implies that ϕ_0 results in the finest representation, whereas ϕ_{π^*} causes the coarsest representation. Moreover, each abstraction level can be seen as a special case of all those abstraction levels that lead to a coarser representation.

When performing an abstraction, one would like to maintain optimality of planning and improve computational efficiency as much as possible. Regarding the impacts of these levels of abstraction on the optimality of planning, [36] lay out that the optimal abstract policy $\overline{\pi}^*$ is optimal in the ground MDP for ϕ_{model}, ϕ_{Q^π}, ϕ_{Q^*}, and ϕ_{a^*}, but that this is not always the case for ϕ_{π^*}. Furthermore, the authors

demonstrate that for ϕ_{model}, ϕ_{Q^π}, ϕ_{Q^*}, and ϕ_{a^*}, model-based RL will converge to a value function whose greedy policy is optimal in the ground MDP, as long as each state in the ground model is given a fixed weight in the abstract model. The same convergence result also holds for Q-learning with a fixed policy. Yet, the convergence of model-based RL and of Q-learning with a fixed policy is not guaranteed for ϕ_{π^*}. Hence, all discussed levels of abstraction but ϕ_{π^*} preserve sufficient information to generally enable optimal planning in the ground MDP. However, the authors' experiments in four domains with 400 to 1,024 states reveal that of the levels of abstraction with guaranteed optimality, only ϕ_{a^*} results in a useful reduction in the size of the state space.

Application to internal state representations of neural networks. These levels of state abstraction do not impose any Euclidean distances between abstract states. Thus, applied to the latent representations of neural networks, they solely consider which states are mapped to *exactly the same* activations in a network layer. Our concept of ideal internal state representation, however, takes the Euclidean distances of the activations of *all* states into consideration. Therefore, when we speak of the coarsest Markov state representation in this work, we mean a state representation in which states are mapped to the same activation pattern if and only if they are mapped to the same abstract state in the coarsest Markov state representation as it is defined in the context of state abstraction, and where Euclidean distances between abstract states are proportional to *how* behaviorally similar those states are. Similarly, with Q^*-irrelevance abstraction we mean the coarsest Q^*-irrelevance abstraction in which the Euclidean distances between abstract states are proportional to the Euclidean distances between the corresponding Q-values. Yet, it is important to keep in mind that depending on the hidden layer size of a neural network, it may not be possible to create such representations exactly. This is, for example, the case for a Q^*-irrelevance abstraction when the hidden layer size is smaller than the number of actions and the Q-values do not have an intrinsic dimensionality[16] lower than the number of actions.

2.6. Stochastic Bisimulation

The notion of *stochastic bisimulation* [21] stems from the field of state abstraction. It groups states into the same equivalence class, or "block", if and only if they have the same reward and the same transition distribution over all other blocks for all actions. In other words, states are considered equivalent if and only if they are mapped to the same abstract state in the coarsest Markov state representation. Such equivalent states are called *bisimilar*. Thus, an abstract representation that is equivalent to the ground representation under the notion of stochastic bisimulation is the result of an exact model-irrelevance abstraction ϕ_{model}.

2.7. Bisimulation Metrics

Based on the notion of stochastic bisimulation for MDPs defined in the previous section, *bisimulation metrics* have been proposed by [12]. In the following, these metrics and ways to compute them are discussed for finite MDPs and for MDPs with infinite state spaces in Section 2.7.1 and Section 2.7.2, respectively. Notice that this work utilizes bisimulation metrics for finite MDPs as well as for MDPs with infinite state spaces, since the latter can be used to represent POMDPs.

2.7.1. Bisimulation Metrics for Finite MDPs

The work of [12] presents two bisimulation metrics[17] that measure how similar the transition and reward functions of two states in a finite MDP are. Stochastic bisimulation can be used like a metric that assigns a distance of 0 to all bisimilar states and a distance of 1 to all other states. Consequently, these bisimulation metrics are perceivable as a quantitative version of stochastic bisimulation in that they assign a distance of 0 only to bisimilar states and that if the parameters of two bisimilar states are altered on a small scale, the metric distance between the two states will stay small.

Specifically, [12] introduce a general form for bisimulation metrics that measure the distance be-

[16] A dataset $\in \mathbb{R}^d$ has an intrinsic dimensionality of $m \leq d$ if m free variables suffice to (approximately) represent the dataset [58].

[17] In fact, the proposed metrics are pseudometrics as two distinct points can have a distance of 0. Based on a pseudometric, one can obtain an equivalence relation by stating that two points are equivalent if they have a distance of 0 [13]. Since [12] simply use the term "metrics," the same naming convention is followed here to avoid confusion.

tween two states s and s':

$$d(s,s') = max_{a\in A}\Big(c_R|R(s,a) - R(s',a)| + c_T d_P\big(P(s,a),P(s',a)\big)\Big), \tag{2.6}$$

where d_P is a probability metric[18]. c_R and c_T are two positive one-bounded constants that determine the weight given to the difference concerning the reward functions compared to the one assigned to the difference between the transition functions. Based on this general form, the authors propose two bisimulation metrics d_{TV} and d_{fix}:

- The bisimulation metric $d_{TV}(s,s')$ is obtained by letting d_P in Equation 2.6 be the total-variation probability semimetric[19] T_{TV}:

$$d_{TV}(s,s') = max_{a\in A}\Big(c_R|R(s,a) - R(s',a)| + c_T T_{TV}\big(P(s,a),P(s',a)\big)\Big). \tag{2.7}$$

  The total-variation distance between the transition functions of states s and s' is:

$$T_{TV}\big(P(s,a),P(s',a)\big) = \frac{1}{2}\sum_{s_i\in S}|P(s_i|s,a) - P(s_i|s',a)|. \tag{2.8}$$

- The bisimulation metric $d_{fix}(s,s')$ is the least fixed point of iteratively solving Equation 2.6 with the Kantorovich distance $T_K(d)$[20] as the probability metric d_P. More precisely:

$$d_{fix} = \coprod_{t\in\mathbb{N}} d_t, \tag{2.9}$$

  where d_0 is the everywhere-zero metric and d_t, $t > 0$, is given by:

$$d_t(s,s') = max_{a\in A}\Big(c_R|R(s,a) - R(s',a)| + c_T T_K(d_{t-1})\big(P(s,a),P(s',a)\big)\Big). \tag{2.10}$$

  $T_K(d)$ depends on a semimetric d on the state space S that assigns a distance of at most 1 to states. In the case of finite MDPs, $T_K(d)$ reduces to the following linear program:

$$T_k(d)(P,Q) = max_{u_1,\,\ldots,\,u_{|S|}}\sum_{i=1}^{|S|}(P(s_i) - Q(s_i))u_i \tag{2.11}$$

$$s.t.\ \ u_i - u_j \leqq d(s_i,s_j)\quad \forall i,j$$
$$0 \leq u_i \leq 1\quad \forall i,$$

  where P and Q are two state probability functions.

d_{fix} assigns a distance of 0 to states if and only if they are bisimilar. Thus, it is this specific bisimulation metric that we ideally want Euclidean distances between states in an internal state representation to be proportional to. d_{TV}, on the other hand, attributes a distance of 0 only to states whose reward and transition functions are equal with respect to all other states in the input MDP, which is a more restrictive version of bisimulation [14]. Hence, $d_{fix} \leq d_{TV}$.

2.7.1.1 Exact Computation

The bisimulation metric d_{TV} for two states can be obtained by simply computing Equation 2.7 a single time, which has a complexity of $O(|A||S|)$. The bisimulation metric $d_{fix}(s,s')$, however, is the least fixed point of iteratively solving Equation 2.10, beginning with d_0 as the everywhere-zero metric. For each iteration $t > 0$, $T_K(d_{t-1})\big(P(s_i,a),P(s_j,a)\big)$ has to be computed for the transition functions $P(s_i,a)$ and $P(s_j,a)$ of all pairs of states $s_i,\ s_j$, $i \neq j$, for each action $a \in A$. This computation can be achieved via a minimum cost flow solver, such as the one described in Section D.3.2 in the Appendix, and requires

[18] A probability metric "is a metric on a suitable set of probability distributions in some measurable space S" [32].
[19] In contrast to a metric, a semimetric does not necessarily satisfy the triangle inequality.
[20] The Kantorovich metric is a probability semimetric and is also called Wasserstein metric or Monge–Kantorovich metric.

a transportation network to be created. To this end, two copies of each state are made, whereby one copy serves as a demand node and the other copy as a supply node. Thereby, the supply of the supply node and the demand of the demand node for state s_k are given by:

$$supply_k = P(s_k|s_i, a),$$
$$demand_k = P(s_k|s_j, a).$$

Moreover, there is an arc from every supply node to every demand node, and this arc has a cost that is equal to the distance between the corresponding states based on d_{t-1}. The resulting transportation network is depicted in Figure 2.1.

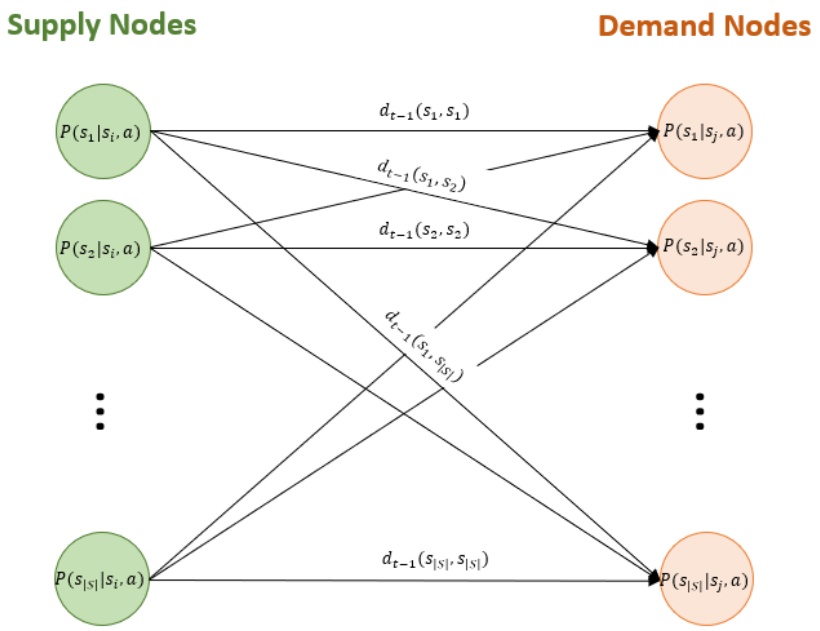


Figure 2.1: The transportation network to compute $T_K(d_{t-1})\big(P(s_i,a), P(s_j,a)\big)$ for the transition functions of states s_i and s_j for action a, whereby arcs are labeled with their costs. $T_K(d_{t-1})\big(P(s_i,a), P(s_j,a)\big)$ is equal to the cost of the flow with minimum cost for this network.

A minimum cost flow solver computes for this transportation network the flow of minimum total cost. A flow thereby is a set of quantities transported across the arcs such that the sum of quantities arriving at a demand node v is equal to its demand and the sum of quantities leaving each supply node u is equal to its supply:

$$supply_k = P(s_k|s_i,a) = \sum_{l=0}^{|S|-1} f(u_k, v_l) \qquad \forall\ k \in \{0,\ \ldots,\ |S|-1\}$$
$$demand_k = P(s_k|s_j,a) = \sum_{l=0}^{|S|-1} f(u_l, v_k) \qquad \forall\ k \in \{0,\ \ldots,\ |S|-1\},$$

where $f(u_k, v_l)$ is the quantity transported from the supply node of state s_k to the demand node of state s_l. The cost of such a flow is the sum of the products of the quantities transported across arcs and the corresponding arc costs:

$$\sum_{k,l \in \{0,\ \ldots,\ |S|-1\}} f(u_k, v_l) d_{t-1}(s_k, s_l). \tag{2.12}$$

$T_K(d_t)\big(P(s_i,a), P(s_j,a)\big)$ then is equal to the cost of the flow with minimum total cost.

The pseudocode for computing d_{fix} based on solving these transportation networks is shown in Algorithm 1. Precisely calculating d_{fix} is infeasible for problems with large state spaces, as the runtime

Algorithm 1 Computation of $d_{fix}(s_i, s_j)\ \forall s_i, s_j \in S$

Require: $\langle S, A, R, P, \gamma \rangle$ with R and P as matrices, precision λ
$d \leftarrow [0]$ {Initialize pairwise distances of all states to 0}
$T \leftarrow \left\lceil \frac{ln(\lambda)}{ln(\gamma)} \right\rceil$ {Number of iterations to guarantee precision λ}
for $t \leftarrow 0$ to $T-1$ **do**
  $d_t \leftarrow d$
  for $i \leftarrow 0$ to $|S|-2$ **do**
    for $j \leftarrow i+1$ to $|S|-1$ **do**
      $dist_{max} \leftarrow 0$ {Largest distance between s_i and s_j for any $a \in A$}
      for $a \in A$ **do**
        $T_K \leftarrow T_K(d)(P[s_i, a], P[s_j, a])$ {Computed with min. cost flow solver}
        $dist_{max} \leftarrow max(dist_{max}, (1-\gamma)|R[s_i, a] - R[s_j, a]| + \gamma T_K)$
      end for
      $d_t[s_i, s_j] \leftarrow dist_{max}$
      $d_t[s_j, s_i] \leftarrow dist_{max}$
    end for
  end for
  $d \leftarrow d_t$
end for
return d

complexity of computing d_{fix} with a guaranteed precision of λ is $O\big(|A||S|^4 log|S| \frac{ln\ \lambda}{ln\ c_T}\big)$. This complexity stems from determining the minimum cost flow of a transportation network during each iteration t for all pairs of states and for each action $a \in A$. Solving the transportation network thereby can be achieved in $O(|S|^2 log\ |S|)$. Furthermore, $\left\lceil \frac{ln\ \lambda}{ln\ c_T} \right\rceil$ iterations guarantee that d_{fix} is calculated with a precision of λ.

2.7.1.2 Other Approaches to Compute d_{fix}

Besides the original exact algorithm by [12] to compute d_{fix} discussed in the previous section, several other algorithms have been proposed that address some of the issues of the exact algorithm. For example, the on-the-fly methods by [9] target the problem that the exact algorithm compares all pairs of states at every iteration and guarantees convergence solely if those comparisons are performed exactly. Other approximate algorithms for calculating d_{fix} are the statistical sampling approach by [14], the asynchronous algorithm by [4], the extension to [4] that makes use of approximants exploiting the state space structure by [5], and the approach for deterministic MDPs using neural networks by [8]. In our work, we solely make use of the approximation algorithm by [8] besides the original exact algorithm by [12]. We employ the approximation approach by [8], because it is applicable to both continuous and discrete MDPs. Thus, it can also be utilized to efficiently approximate d_{fix} based on the belief states corresponding to action-observation histories from deterministic POMDPs. This algorithm therefore is briefly discussed next.

Approximation of d_{fix} via neural networks. In [8], the authors present an approach to calculating d_{fix} for deterministic MDPs with both finite and continuous state spaces via neural networks. We use this algorithm to efficiently approximate d_{fix} for MDPs as well as based on belief states for POMDPs. The proposed method is based on sampling transition pairs $\{\langle s, a, R(s,a), N(s,a)\rangle, \langle s', a, R(s',a), N(s',a)\rangle\}$ from a distribution D, where $N(s,a)$ is the deterministic next state after taking action a in state s. For example, one may sample transitions uniformly at random from a replay memory. The authors show that if one samples such transition pairs, an iterative procedure starting with d_0 as the everywhere-zero metric and with the following update converges to d_{fix} almost surely as $t \to \infty$ in the tabular case[21]:

$$d_t(s_i, s_j) \leftarrow \begin{cases} max\big(d_{t-1}(s_i, s_j), |R(s_i, a) - R(s_j, a)| \\ \quad +\gamma d_{t-1}(N(s_i, a), N(s_j, a))\big) & if\ s_i = s,\ s_j = s', \\ d_{t-1}(s_i, s_j) & otherwise. \end{cases}$$

[21] Refer to Theorem 4 in [8] for the proof.

Thus, after sampling a transition pair $\{\langle s, a, R(s,a), N(s,a)\rangle, \langle s', a, R(s',a), N(s',a)\rangle\}$ for iteration t, only the distance between s and s' is updated. For all other pairs of states, the distance from iteration $t-1$ is copied.

Based on this finding, the authors propose an approximation algorithm for d_{fix} that uses neural networks. This approximation approach makes use of an online network[22] ψ_θ with parameters θ and a target network ψ_{θ^-} with parameters θ^-. Both ψ_θ and ψ_{θ^-} take as input the concatenated state representations of two states s and s' and output an approximation to $d_{fix}(s,s')$. The target objective at iteration t for a sampled transition pair $\{\langle s, a, R(s,a), N(s,a)\rangle, \langle s', a, R(s',a), N(s',a)\rangle\}$ is as follows if $s \neq s'$:

$$T_{\theta_t^-}(s,s',a) = max\Big(|R(s,a) - R(s',a)| + \gamma\psi_{\theta_t^-}([\phi(N(s,a)), \phi(N(s',a))]), \psi_{\theta_t^-}([\phi(s), \phi(s')])\Big), \quad (2.13)$$

where $\phi : S \to \mathbb{R}^k$ is a k-dimensional representation of the state space and [.] denotes concatenation. Whenever $s = s'$, the target objective is equal to 0.

For a single transition pair, the loss $L_{s,s',a}$ is:

$$L_{s,s',a} = E_D\Big(T_{\theta_t^-}(s,s',a) - \psi_{\theta_t}([\phi(s)], \phi(s'))\Big)^2. \quad (2.14)$$

In our work, however, we make use of mini batches of b transitions for the approximation algorithm. In that case, the authors of [8] specify the analogous loss as so:

$$L_t(\theta_t) = \mathbb{E}_D\Big[W \otimes \big(T - \psi_{\theta_t}(S^2)\big)^2\Big], \quad (2.15)$$

where $S^2 \in \mathbb{R}^{b\times b\times 2k}$ contains the concatenations of the representations of all pairs of states in the mini batch, W is a mask that ensures that the actions for pairs of transitions are the same, $\otimes$ stands for the Hadamard product[23], and $\psi(X)$ denotes that ψ is applied to each element of matrix X. Furthermore, the matrix T is defined as follows:

$$T = (1 - I) \otimes max\Big(R^2 + \gamma\beta\psi_{\theta_t^-}(N^2), \beta\psi_{\theta_t^-}(S^2)\Big), \quad (2.16)$$

where I is the identity matrix, $N^2 \in \mathbb{R}^{b\times b\times 2k}$ contains the concatenations of the representations of all pairs of next states in the mini batch, $R^2 \in \mathbb{R}^{b\times b}$ is a matrix with the absolute reward differences, and β is a stability parameter that is initialized to 0 and incremented towards a maximum of 1 every time the target network is updated. The multiplication by $(1 - I)$ ensures that exact targets of 0 are given for $d(s,s)$ rather than approximations. Notice thus that the only difference between Equations 2.14 and 2.15, besides the use of multiple transition pairs for a single update in the latter, is that the stability parameter β is added when mini batches are used. The pseudocode for the computation of the approximation d'_{fix} to d_{fix} for all $s, s' \in S$ via the proposed algorithm is given in Algorithm 2.

2.7.2. Bisimulation Metrics for MDPs with Infinite State Spaces

In [15], the notion of bisimulation metrics from [12, 14] is extended to MDPs with infinite state spaces, which include MDPs with continuous state spaces. Under the assumption that the rewards are bounded and vary continuously and that the transition probabilities are continuous, [15] show that the bisimulation metrics are also continuous in the transition and reward probabilities and bound the optimal value of states in a continuous fashion. While our work does not directly use MDPs with infinite state spaces, we compute bisimulation metrics for the belief states of POMDPs. Recall that a POMDP can be regarded as a continuous MDP whose states are belief states. To calculate d_{fix} based on belief states, we employ the approximation algorithm by [8] that is described in the previous section in the context of finite MDPs. The reason for utilizing this approximation algorithm is that computing d_{fix} exactly faces similar runtime challenges as in the case of finite MDPs. Furthermore, a Monte Carlo approximation scheme for d_{fix} by [13] is not scalable to large domains.

[22] The term *online network* is used to denote that the parameters of this network are updated each iteration, whereas those of the *target network* are updated solely every c iterations. The reason for utilizing an online and a target network is to foster learning stability just as in the context of deep Q-learning (see Section 2.4.1).

[23] The Hadamard product of two matrices A and B with the same dimensions is a matrix C such that $C_{i,j} = A_{i,j} \cdot B_{i,j}$.

Algorithm 2 Computation of $d'_{fix}(s, s')\ \forall s, s' \in S$

Require: Replay memory M with transitions $\{\langle s, a, R(s,a), N(s,a)\rangle\}$, discount factor γ, online network ψ_θ, target network ψ_{θ^-}, number of episodes T, target network update frequency c, increment for stability parameter β_{inc}, batch size b, state representation $\phi : S \to \mathbb{R}^k$

$\beta \leftarrow 0$ {Initialize stability parameter}
for $t \leftarrow 0$ to $T-1$ **do**
  if $t \,\%\, c = 0$ **then**
    $\beta \leftarrow max(1, \beta + \beta_{inc})$
    $\theta^- \leftarrow \theta$ {Update target network}
  end if
  $B \leftarrow$ sample of b transitions from M
  $S^2 \leftarrow$ concatenations of representations of all pairs of states in B
  $N^2 \leftarrow$ concatenations of representations of all pairs of next states in B
  $t_1 \leftarrow \beta\psi_{\theta^-}(S^2)$
  $R^2 \leftarrow$ absolute reward differences $|R(s,a) - R(s',a)|$ for all pairs of states s, s' in B
  $t_2 \leftarrow R^2 + \gamma\beta\psi_{\theta^-}(N^2)$
  $T \leftarrow (1-I) \otimes max(t_1, t_2)$ {I is identity matrix}
  $W \leftarrow$ action mask to ensure matching actions for transition pairs
  $output\_curr \leftarrow \psi_\theta(S^2)$
  $L \leftarrow MSEloss(W \otimes T, W \otimes output\_curr)$
  $\theta \leftarrow$ update of θ based on L
end for
$d'_{fix} \leftarrow 0$ {Matrix with pairwise distances of states}
for $i \leftarrow 0$ to $|S| - 2$ **do**
  for $j \leftarrow i+1$ to $|S| - 1$ **do**
    $d'_{fix}[i,j] \leftarrow \psi_\theta([\phi(s_i), \phi(s_j)])$
    $d'_{fix}[j,i] \leftarrow d'_{fix}[i,j]$
  end for
end for
return d'_{fix}

2.8. t-Distributed Stochastic Neighbor Embedding

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear technique for visualizing high-dimensional data in a two- or three-dimensional space that can capture both local and global structure such as clusters [40]. The ability to maintain both local and global information is an important difference to other dimensionality reduction techniques. For example, linear dimensionality reduction methods such as *Principal Component Analysis* (PCA) [26] focus on preserving large distances, and other non-linear approaches such as Sammon mapping [54] can maintain the local but not the global structure [40]. We employ the t-SNE algorithm to visualize the internal state representations learned by deep RL agents.

t-SNE operates by minimizing discrepancies between similarities in the original high-dimensional space and similarities in the low-dimensional space. First, Euclidean distances between data points in the original high-dimensional space are converted to similarities. Specifically, the symmetric similarity p_{ij} of datapoints x_i and x_j is given by

$$p_{ij} = p_{ji} = \frac{p_{i|j} + p_{j|i}}{2n},$$

where n is the total number of datapoints. This symmetric similarity is a combination of the asymmetric similarities $p_{i|j}$ and $p_{j|i}$ to ensure that the low-dimensional locations of outliers in the high-dimensional space also impact the cost function. The asymmetric similarity $p_{j|i}$ of datapoint x_j to datapoint x_i in the high-dimensional space thereby is given by:

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}.$$

Hence, $p_{j|i}$ is the probability that x_j is selected as the neighbor of x_i if neighbors are chosen based on their probability density under a Gaussian that is centered at x_i and has variance σ_i. Each σ_i thereby is determined via binary search such that the following holds:

$$Perp(P_i) = 2^{H(P_i)},$$

where P_i is the probability distribution over all other datapoints induced by σ_i, $Perp(P_i)$ is the perplexity chosen by the user and $H(P_i)$ is the bit-based entropy of P_i. The perplexity is perceivable as a smooth measure of the effective number of neighboring datapoints taken into consideration. Common settings for the perplexity are values between 5 and 50, but the algorithm is rather insensitive to different choices [40]. In our experiments, we use a default perplexity value of 30, but also utilize some other values to verify the patterns we observe.

Second, the Euclidean distance between the low-dimensional locations y_i and y_j of datapoints x_i and x_j is transformed to a symmetric similarity as so:

$$q_{ij} = q_{ji} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}.$$

Hence, instead of a Gaussian distribution, a Student-t distribution with one degree of freedom is utilized to compute similarities in the low-dimensional space. The reason for this is that the heavy-tailed character of the Student-t distribution helps to ameliorate the crowding problem. This problem arises, because an insufficient amount of area is available in a two-dimensional space to depict moderately far apart datapoints compared to the amount of area that is available for visualizing nearby datapoints. In addition, the density of a point under a Student-t distribution can be computed much quicker than under a Gaussian distribution due to the lack of an exponential.

Based on the aforementioned similarities, t-SNE uses gradient descent to minimize the *Kullback-Leibler* (KL) divergence between the joint probability distributions P and Q:

$$KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}.$$

To further improve the visualizations, gradient descent is supplemented by *early compression* and *early exaggeration*. The former adds an L_2-penalty to the objective function that is proportional to the sum

of squared distances of the low-dimensional points from the origin, and thus compels datapoints in the low-dimensional space to remain close together at the beginning. Early exaggeration, on the other hand, is achieved by multiplying all p_{ij} values by some factor $r > 1$ such as 4 during the first iterations. This causes clusters in the data to form widely separated clusters in the low-dimensional space. Since this objective function is not convex, multiple optimization parameters need to be determined and the outcome of the algorithm can be different for each execution. However, experimental results on a variety of datasets suggest that the same optimization parameter settings can be appropriate for several datasets and that the final KL divergence does not differ much between runs [40].

2.8.1. Using t-SNE to Interpret Data

While t-SNE is very useful to analyze structures in high-dimensional data, there are several aspects one has to pay attention to when deriving insights from t-SNE plots [63]:

1. For some datasets, successive executions of the algorithm do not always lead to the same result. The authors of [63] show that different runs yield different global shapes for data in the shape of a trefoil knot, but that similar outcomes are obtained for more simple datasets.

2. The algorithm strives to obtain roughly equally sized clusters, which means that it may expand denser clusters and contract less dense ones.

3. Seeing correct distances between clusters in a t-SNE plot makes it necessary to fine-tune the perplexity value or may not be possible at all. Even clusters that have very different distances between them in the original high-dimensional space may look equidistant in a t-SNE plot.

4. Not all clusters in a t-SNE plot are meaningful. Even if the data to be visualized is drawn from a high-dimensional unit Gaussian distribution, distinct clusters may appear in t-SNE plots.

Thus, it is good practice to try a few different perplexity values and to run the algorithm multiple times with the same hyperparameter settings to get an idea of the reproducibility of the algorithm's result. Section C.6 in the Appendix exemplarily shows the impact of different perplexity values on the t-SNE plot of the activations states are mapped to in the hidden layer of a 2-layer DQN.

3

Characteristics of Internal State Representations During Learning

The black-box nature of neural networks makes it hard to comprehend what deep RL agents learn and whether this is in line with what we ideally would want them to learn. Intuitively, a deeper understanding of what such agents learn should thus enable us to comprehend to a higher degree why certain algorithms or network architectures perform better than others and to develop methods that specifically target discrepancies between what is and what should be learned. When we speak of what a deep RL agent learns, we thereby mean the internal state representations that it forms of its environment. Such an internal state representation is the combination of all distinctions the agent makes for the observations it receives from its environment and may consist of multiple internal states. Each observation (history) from the environment is mapped to exactly one internal state. For example, if a firefighter robot has learned to not distinguish observations based on the color of a house, it may map the observations ”smoke above blue house” and ”smoke above orange house” to the internal state ”smoke above house” and the house color thus cannot inform the agent’s action choice. In a neural network, an internal state representation exists in each network layer as the combination of the activations (histories of) observations are mapped to.

What should a deep RL agent ideally learn? To minimize the amount of data, time and memory required for training, an agent should ideally form an internal state representation that is as small as possible while still allowing the agent to learn to act optimally in a domain. In addition, the learned internal state representations should at test time enable an agent to generalize to new values of irrelevant features and to quickly adapt to modifications to the rewards or transition probabilities, as long as these modifications do not make previously irrelevant features relevant. These two aspects are important, because high-dimensional observations such as images render it infeasible to train an agent with all possible feature values, and some applications, such as in robotics, may feature domain shifts. Lastly, since rewards and transition probabilities are commonly approximated, the representation should be relatively insensitive to small estimation errors. A state representation that satisfies these criteria is the *coarsest Markov state representation* and it is hence this specific state representation that lies at the heart of our work[1]. In this representation, Euclidean distances between states are proportional to how behaviorally similar the states are, which is measured by the bisimulation metric d_{fix}[2].

Why may a deep RL agent not learn what it should ideally learn? Practically, however, an agent may not learn the coarsest Markov state representation. This is due to several reasons, among which are the following:

[1] Recall that our definition of the coarsest Markov state representation differs from the one used in the context of state abstraction. In state abstraction, the coarsest Markov state representation merely is the representation that regards states as equivalent if and only if they are bisimilar and thus does not impose any distances between internal states. See Section 2.1 for details on the coarsest Markov state representation as it is defined in our work.

[2] Refer to Section 2.7.1 for more information on this bisimulation metric.

- *The state space may be small compared to the capacity of the agent.* In this case, the agent can precisely store relevant information for each single state, which renders any abstraction of the state space unnecessary. Even if some abstraction occurs, one state equivalence class under the notion of stochastic bisimulation could be distributed over multiple internal states.
- *The output layer of a Q-learning agent has to group states together that have the same expected cumulative reward for taking any action $a \in A$ and executing an optimal policy afterwards.* In this form of abstract state representation called Q^*-irrelevance abstraction, even states that are not bisimilar may be considered equivalent[3]. This also implies that the representations learned in hidden layers might just fall between the encoding of the input and a Q^*-irrelevance abstraction.

The purpose of the experiments in this chapter therefore is to explore the internal state representations formed in the network layers of deep RL agents during training. Thereby, we are especially interested in how similar to the coarsest Markov state representation the internal state representations are. To this end, we analyze the process by which the internal state representations are learned and the factors that impact this process in Section 3.2. All of our experiments in this chapter are performed for model-free Q-learning agents for *fully* observable domains, but initial results for *partially* observable domains are provided in Chapter B in the Appendix. Note that while we investigate the internal state representations in both hidden and output layers, especially the hidden-layer representations are of interest, as the output layer of a Q-learning agent is constrained to learn to represent the Q-values. Our experimental approach is laid out in more detail in Section 3.1.

3.1. Methodology

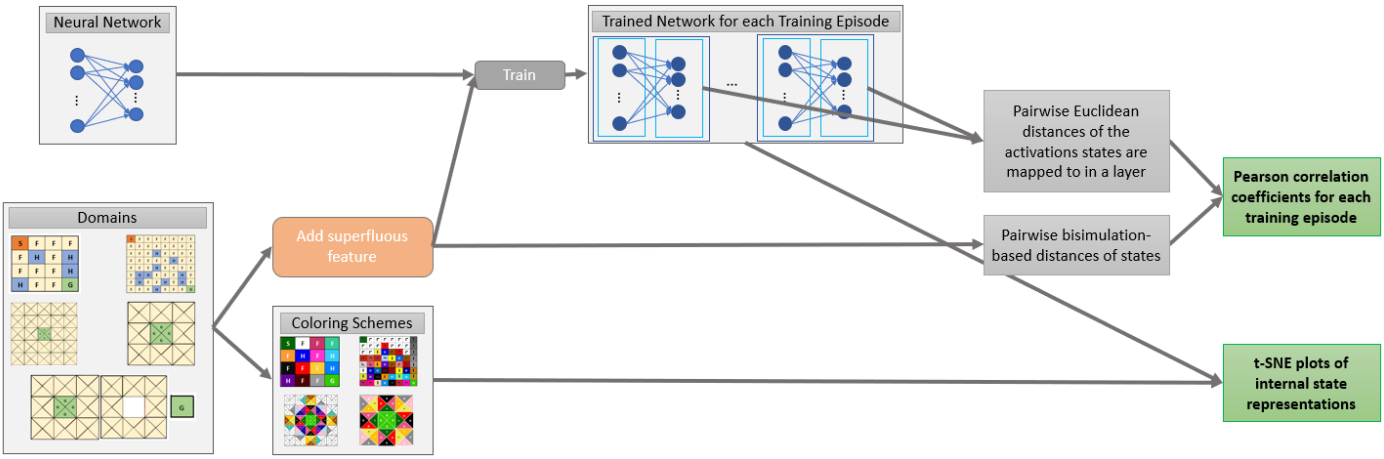


Figure 3.1: Flow-chart of the experiments we conduct for a neural network for the first research question. The actions colored in orange are performed solely for fully observable domains.

We train neural networks for approximately representing Q-functions as described in Section 2.4.1 for several fully and partially observable domains. For the fully observable domains, we thereby add a superfluous feature to the observations so that the observations contain more information than is necessary. For our experiments, we use different network architectures and types of state encoding to explore the impact of these factors on the learning process. For each network architecture, state encoding and domain, we compute bisimulation-based correlation coefficients for each training episode and network layer. These correlation coefficients measure how similar to the coarsest Markov state representation and more specifically how Markov with respect to the rewards and transitions an internal state representation is[4]. We also create t-SNE plots[5] of internal state representations to intuitively

[3] When we use the term Q^*-irrelevance abstraction in our work, we mean the coarsest Q^*-irrelevance abstraction in which the Euclidean distances between abstract states are proportional to the Euclidean distances between the corresponding Q-values. Note that a Q^*-irrelevance abstraction as defined in the context of state abstraction does not impose any Euclidean distances between abstract states. Refer to Section 2.5 for more information on a Q^*-irrelevance abstraction in state abstraction.

[4] A state representation that is *Markov with respect to the reward* is one in which knowledge of past states does not allow for a more accurate prediction of the next reward [42]. The definition of *Markov with respect to the transitions* proceeds analogously.

[5] Refer to Section 2.8 for more information on the t-SNE algorithm.

visualize what an agent has learned. A flow-chart for the set of experiments conducted for a neural network for this research question is depicted in Figure 3.1. Subsequently, we provide detailed information on the computed correlation coefficients, domains, state encoding, network architectures and training procedure, and the t-SNE visualizations that we create for the internal state representations learned by agents in *fully* observable domains. Our precise methodology and results for *partially* observable domains are discussed in Chapter B in the Appendix.

3.1.1. Correlation Coefficients

The Pearson correlation coefficient is a measure of the linear correlation between two variables. We calculate Pearson correlation coefficients based on bisimulation metrics as well as some other measures on the one hand and the Euclidean distances between the activations states are mapped to in a network layer on the other hand. Letting $d_{eucl}(s_i, s_j)$ denote the Euclidean distance of the activations s_i and s_j are mapped to and $d_m(s_i, s_j)$ the distance of s_i and s_j for some, in our case typically bisimulation-based, measure, the Pearson correlation coefficient r_{d_{eucl},d_m} is:

$$r_{d_{eucl},d_m} = \frac{\sum_{i=0}^{|S|-2}\sum_{j=i+1}^{|S|-1}(d_{eucl}(s_i,s_j) - \overline{d_{eucl}})(d_m(s_i,s_j) - \overline{d_m})}{\sqrt{\sum_{i=0}^{|S|-2}\sum_{j=i+1}^{|S|-1}(d_{eucl}(s_i,s_j) - \overline{d_{eucl}})^2}\sqrt{\sum_{i=0}^{|S|-2}\sum_{j=i+1}^{|S|-1}(d_m(s_i,s_j) - \overline{d_m})^2}}, \tag{3.1}$$

where $\overline{\cdot}$ denotes an average. Note that $\frac{|S|(|S|-1)}{2}$ samples are used for the computation.

3.1.1.1 Correlation Coefficients Based on Bisimulation Metrics

We compute correlation coefficients based on both bisimulation metrics and bisimulation metric components. The reasoning for also calculating component measures is twofold. First, the bisimulation distance between two states is a weighted sum of the distance of their reward functions and the distance of their transition functions. The weights for these two components typically are set to $1-\gamma$ and γ, respectively [12]. Yet, the discount factor γ is in some sense arbitrary, as many domains have no fixed γ associated with them and a neural network may learn successfully for multiple different values for γ. Thus, by calculating the component measures rather than the composed bisimulation metric, the problem of choosing an appropriate value for γ is avoided. Second, calculating the component measures allows to distinguish when the state representation becomes approximately Markov with respect to the next state from when the state representation approximately allows the prediction of the next reward. When evaluating the results based on these component measures, however, it is important to remember that the predictability of the next reward typically only contributes by a very small amount to the bisimulation metrics as high values for γ are common.

These are the correlation coefficients that are based on *exact* (components of) bisimulation metrics[6]:

1. $c_{K(d_{fix})}$.
   - *Definition*. This is the Pearson correlation coefficient between the bisimulation metric d_{fix} on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. d_{fix} is not only dependent on the transition functions of states, but also on their reward functions and the discount factor γ[7]. As Euclidean distances in the coarsest Markov state representation are proportional to distances assigned by d_{fix}, $c_{K(d_{fix})}$ is a measure of whether a state representation is similar to the coarsest Markov state representation.
   - *Computation*. Computing $c_{K(d_{fix})}$ requires the calculation of d_{fix} as the least fixed point of Equation 2.6 with $d_P = T_K(d)$. We compute this least fixed point exactly for all our domains[8].

[6] Additionally, we originally computed another bisimulation-based correlation coefficient, c_K, which proved not to be useful and is described in Section D.1.2 in the Appendix.

[7] This dependence on the reward function and the discount factor originally motivated us to compute the correlation coefficient c_K as described in Section D.1.2 in the Appendix.

[8] This computation is expensive. For Gridworld 5x5 with the state space augmented by means of the superfluous feature value and thus a state space size of 500, calculating d_{fix} exactly takes several days on CPUs of the TU Delft HPC cluster. Of course, we could simplify the computations based on our domain knowledge that all states that differ only in the superfluous feature value are bisimilar and hence have the same values for d_{fix} for all other states, but such domain knowledge may not be available in practice.

The pseudocode for the computation of d_{fix} is given in Algorithm 1 in Section 2.7.1.1. Afterwards, the Pearson correlation coefficient is determined as defined in Equation 3.1.

2. c_{Rew}.

   - *Definition*. This is the Pearson correlation coefficient between the maximum absolute reward distances of the states on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. Note that the absolute reward distance of states is a component of the bisimulation metrics (see Equation 2.6). A high value for c_{Rew} indicates that there is one cluster in the space of activations for each set of states that have the same immediate rewards for all actions, and that the Euclidean distances between those clusters correspond to how different the immediate rewards are.
   - *Computation*. The maximum absolute reward distance between states s_i and s_j is given by:

$$max_{a \in A} |R(s_i, a) - R(s_j, a)|. \tag{3.2}$$

   After the calculation of the pairwise maximum absolute reward distances between states and the pairwise Euclidean distances of the activations states are mapped to, the Pearson correlation coefficient is computed (see Equation 3.1).

3. c_{TV}.

   - *Definition*. This is the Pearson correlation coefficient between the maximum total-variation distances T_{TV} of the transition functions of the states on the one hand (see Equation 2.8) and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. Recall that the total-variation distance of the transition function of states is a component of the bisimulation metric d_{TV} (see Equation 2.7). A high value for c_{TV} indicates that there exist clusters in the latent space for all states that have the same transition probabilities for all actions, and that the Euclidean distances between those clusters are proportional to how different the transition probabilities are[9]. c_{TV} can thus be seen as a measure of how well a state representation allows for the prediction of the next state in the input MDP[10].
   - *Computation*. Computing T_{TV} for a pair of states s_i, s_j and an action a requires summing the absolute distances between the probabilities of transitioning from s_i and s_j to each state after taking a. Afterwards, the maximum over all actions is taken when calculating c_{TV}. The pseudocode for this is given in Algorithm 3 and the subsequent computation of the Pearson correlation coefficient is defined in Equation 3.1.

Algorithm 3 Computation of $max_{a \in A}\left(T_{TV}\left(P(s_i, a), P(s_j, a)\right)\right)$ for $s_i, s_j \in S$

Require: S, A, $P(s_i)$ (transition function of s_i $\forall a \in A$), $P(s_j)$ (transition function of s_j $\forall a \in A$)

```
T_TV_max ← 0
for a ∈ A do
    T_TV_max ← max(T_TV_max, ½ Σ_{s∈S} |P(s|s_i, a) − P(s|s_j, a)|)
end for
return T_TV_max
```

We calculate these (components of) bisimulation metrics based on the exact tabular reward and transition functions of a domain. However, Section C.2 in the Appendix also analyzes the impact of using empirical reward and transition functions for c_{Rew} and c_{TV} instead.

Furthermore, we compute a correlation coefficient based on *approximate* bisimulation metrics for some experiments:

[9] Since T_{TV} is a probability semimetric, it does not necessarily fulfill the triangle inequality. Therefore, it may not be possible to create a state representation in which *all* Euclidean distances between the activations of states are *precisely* proportional to T_{TV}.

[10] The *next state in the input MDP* is not necessarily the same as the *next internal state* of an internal state representation.

4. $c_{d'_{fix}}$.

   - *Definition*. This is the coefficient of the correlation between the distances of the states with respect to an approximation of d_{fix} on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. We compare using $c_{d'_{fix}}$ to utilizing $c_{K(d_{fix})}$ in Section C.3 in the Appendix, since d_{fix} is very costly to exactly compute in practice.
   - *Computation*. d_{fix} is approximated by means of the algorithm by [8] described in Section 2.7.1.2[11]. The precise hyperparameter values used for the approximation algorithm are given in Section D.1.1 in the Appendix and the computation of the Pearson correlation coefficient is defined in Equation 3.1.

3.1.1.2 Other Correlation Coefficients

For some experiments, we also compute the following correlation coefficients, which are not based on bisimulation metrics:

1. c_E. This is the coefficient of the correlation between the Euclidean distances of the encoded states on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. In domains where states with similar encodings are not behaviorally similar, one would expect c_E to be lower at the end than at the start of training if a representation that is similar to the coarsest Markov state representation is learned.

2. c_{Q^*}. This is the Pearson correlation coefficient between the maximum absolute distances of states with respect to Q^* on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. The maximum absolute distance with respect to Q^* for two states s_i and s_j is given by:

$$max_{a \in A} |Q^*(s_i, a) - Q^*(s_j, a)|. \tag{3.3}$$

   This correlation coefficient is computed to determine how close to a Q^*-irrelevance abstraction[12] the learned state representation of a deep RL agent is. However, notice that calculating c_{Q^*} exactly for large domains is intractable, as it requires knowing the Q-values.

3.1.2. Domains

Our primary experiments are conducted on four fully observable domains, which are the deterministic 4x4 and 8x8 FrozenLake domains from OpenAI Gym and two of the Gridworld domains used in [14]. For each of these domains, one noise dimension is added to the observations to obtain observations with redundant information. This superfluous feature is uniformly at random given one of five possible values[13].

3.1.2.1 Deterministic FrozenLake

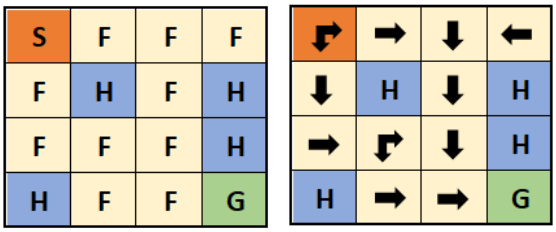
The deterministic FrozenLake domains from OpenAI Gym consist of a grid that contains a start state, a goal state, frozen states and holes. Stepping into a hole as well as exceeding 200 steps or reaching the goal state end an episode. The agent can choose from the actions $\{North, South, East, West\}$ at each non-terminal state. State transitions are deterministic. In the original implementation, a reward of 1 is received for reaching the goal state, and a reward of 0 for all other state-action combinations. To obtain more diverse values for the reward distances of states, the implementation is adapted for this work in that a reward of -1 is given for falling into a hole.

[11] Note that while the algorithm by [8] is designed only for deterministic MDPs, our fully observable domains are stochastic only in the sense that the superfluous feature value of a state is sampled randomly. Thus, the superfluous feature can be seen as noise just as in the experiments conducted by [8] on a 31-state MDP.

[12] Recall that when we use the term Q^*-irrelevance abstraction in our work, we mean the coarsest Q^*-irrelevance abstraction in which the Euclidean distances between states are proportional to the Euclidean distances between the corresponding Q-values.

[13] More information on the superfluous feature is provided in Section 3.1.3.

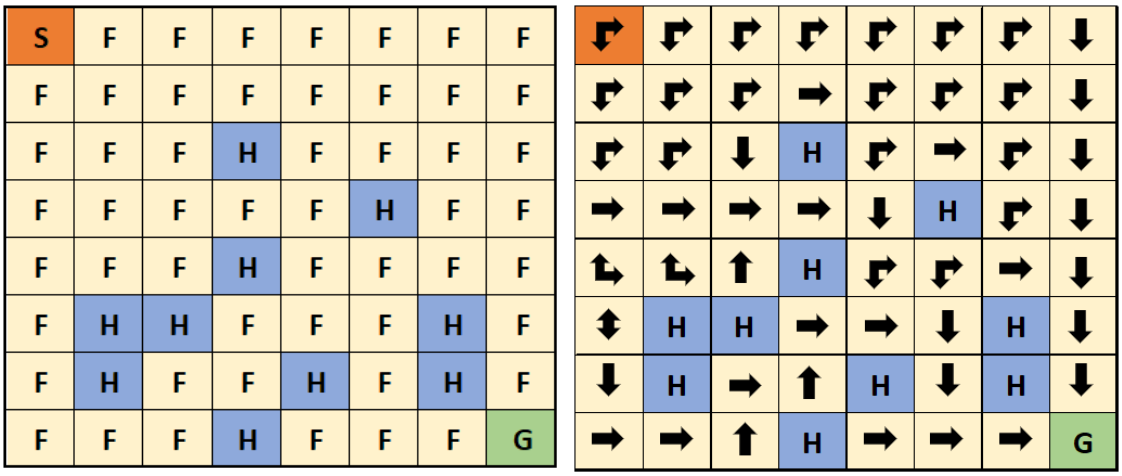
FrozenLake 4x4. The deterministic 4x4 FrozenLake domain consists of a 4x4 grid as depicted in Figure 3.2a. Since there are 11 non-terminal ground states and 5 values for the superfluous feature, a network has found an optimal policy when 55 optimal actions have been learned. Yet, as the start state is fixed and any optimal path from the start to the goal state covers only 6 non-terminal states, learning 30 optimal actions can already mean that an agent always reaches the goal state (see Figure 3.2b).



(a) (b)

Figure 3.2: a) Deterministic 4x4 FrozenLake domain. The start state is denoted by *S*, the goal state by *G*, frozen states by *F*, and holes by *H*. b) Optimal actions for each non-terminal state.

FrozenLake 8x8. The deterministic 8x8 FrozenLake domain consists of an 8x8 grid as depicted in Figure 3.3a. An optimal policy has been achieved when 265 optimal actions have been learned, as there are 53 non-terminal ground states and 5 possible values for the superfluous feature. Yet, since the shortest paths from the start to the goal state require solely 14 actions, learning 70 optimal actions can already mean that an agent always arrives at the goal state (see Figure 3.3b).



(a) (b)

Figure 3.3: a) Deterministic 8x8 FrozenLake domain from OpenAI Gym. The start state is denoted by *S*, the goal state by *G*, frozen states by *F*, and holes by *H*. b) Optimal actions for each non-terminal state.

3.1.2.2 Gridworld

Gridworld 3x3 and 5x5. The Gridworld domains used by [14] consist of a 3x3 or 5x5 grid, whereby the state is a combination of the agent's position on the grid and its orientation (see Figure 3.4). The ground state space thus has a dimensionality of 36 for the 3x3 Gridworld, for instance. The agent can choose from the actions $\{forward, rotate\}$ at each time point. $rotate$ changes the agent's orientation clockwise and $forward$ deterministically moves the agent one step forward if possible. A reward of 1 is obtained for reaching the goal grid location in the center of the grid and a reward of 0 for all other state-action combinations. The start state is chosen uniformly at random from all non-terminal states and performing 100 actions or reaching a goal state end an episode. We implemented these Gridworld domains in the GridWorldSingle package.

Gridworld 3x3 (Aug). Since it holds for the FrozenLake and Gridworld domains that states are bisimilar if and only if they have the same Q-values, we perform experiments with an augmented Gridworld

3x3 domain in which states with the same Q-values are not necessarily bisimilar. Specifically, we copy all non-terminal states of the Gridworld 3x3 domain and create an additional terminal state, resulting in a ground state space dimensionality of 69. For each state copy, the transition and reward probabilities for the optimal action are equal to the ones of the corresponding original state. Taking a non-optimal action a in the copy of state s, however, leads to a reward of $Q^*(s,a)$ and a transition to the new terminal state. Thus, the $Q^*(s,a)$-values of the copied states are equal to the ones of the original states for all actions, but the transition probabilities differ. Note that a Q^*-irrelevance abstraction would map each copied state to the same activation as the corresponding original state, whereas the coarsest Markov state abstraction would map an original state and its copy to distinct activations. This augmented Gridworld 3x3 domain is referred to as Gridworld 3x3 (Aug).

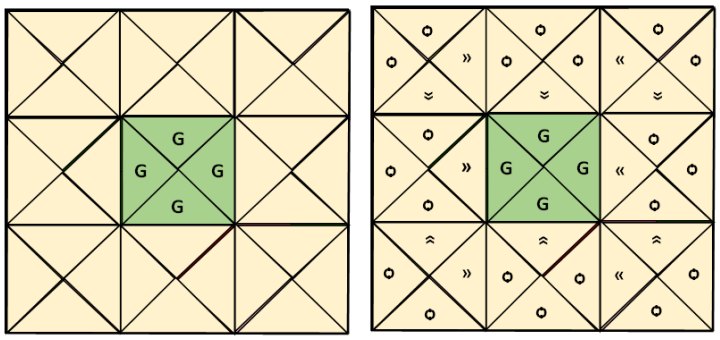


Figure 3.4: Gridworld 3x3 domain. a) Environment grid. For each grid location, four triangles represent the possible orientations of the agent. The start state is chosen uniformly at random from all non-terminal states and the goal states are denoted by *G*. b) Optimal actions for all non-terminal states. Arrows indicate that the optimal action is to move forward, whereas a circle signifies that the optimal action is to rotate clockwise.

3.1.3. State Encoding

We make use of different forms of state encoding, the reason for which is twofold. First, research in the context of classification suggests that the classification performance of a neural network strongly depends on the pre-processing of the input data, including the scaling and encoding of categorical and continuous variables [10, 16, 50]. Thus, the state representation a deep RL learns may also be contingent on the way the states are encoded. Second, each form of state encoding has an inherent correlation with respect to the bisimulation-based measures based on whether states that are close with respect to a measure are encoded in similar ways. This may impact whether the coarsest Markov state representation is learned. Intuitively, encoding bisimilar states in similar ways may make it more likely that a DQN creates the coarsest Markov state representation in one of its layers than one-hot encoding all states, for example.

Ground state encoding. The ground state is either one-hot encoded or encoded via features. These two forms of ground state encoding are referred to by (OH) and (F), respectively. For the FrozenLake 4x4 domain, a ground state is specified via the x- and y-coordinates of the grid location if it is encoded via features. For the Gridworld 3x3 domain, we use the x- and y-coordinates and the orientation as features. For the other domains, we always one-hot encode the ground state. Notice that when the ground states are one-hot encoded, there is a Pearson correlation coefficient of 0 between the Euclidean distances of the encoded ground states and the bisimulation-based measures, whereas this is not necessarily the case for the feature-based encoding. However, one-hot encoding greatly increases the dimensionality of the input and hence is feasible only for moderately-sized state spaces.

Superfluous feature encoding. In addition to these two forms of encoding the ground states, we perform experiments with three different ways of encoding the superfluous feature:

1. The superfluous feature is uniformly at random given one of 5 possible values between 0 and 4. Especially in the case of one-hot encoded states, this leads to a significantly different scale for the superfluous feature compared to the features encoding the ground state. Since the feature

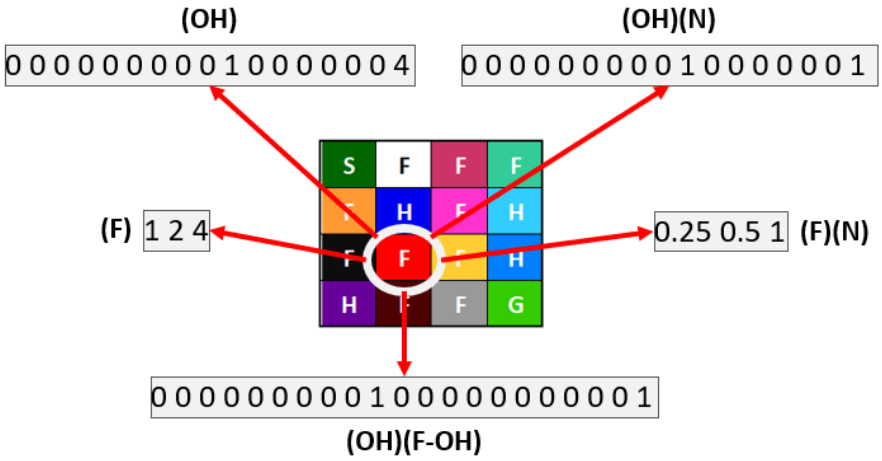


Figure 3.5: Different ways of encoding ground state 9 with a superfluous feature value of 4 in the FrozenLake 4x4 domain. **Ground state**: (OH) and (F) signify that the ground state is one-hot encoded and encoded via features, respectively. **Superfluous feature**: No extension means that the superfluous feature is given a value from $\{0, 1, 2, 3, 4\}$, (N) denotes that the superfluous feature and the features encoding the ground state are subsequently scaled to the interval $[0, 1]$, and (F-OH) expresses that the superfluous feature value is one-hot encoded.

values impact the weight updates during gradient descent, this may cause some weights to update faster than others. Yet, it is interesting to observe whether a DQN learns to map states with solely different values for the superfluous feature to similar activations in this case, as with a random weight initialization such states are initially mapped to activations with comparatively large Euclidean distances between them. When one-hot encoding the ground states, for instance, the Euclidean distance between the activations the encoded states $s_1 = [0, 1, 1]$ and $s_2 = [0, 1, 3]$, which belong to the same ground state $[0, 1]$, are mapped to will tend to be larger at the beginning of training than the one between the activations that s_1 and $s_3 = [1, 0, 1]$, which belong to different ground states but have the same superfluous feature value, are mapped to.

2. The superfluous feature is randomly given one of 5 possible values between 0 and 4 and both the features encoding the ground state and the superfluous feature are subsequently scaled to the interval $[0, 1]$[14]. This encoding ensures that the features do not have different scales. Yet, two states that differ merely in the superfluous feature value are already mapped to comparatively similar activations by the initial DQN, which means that the network has to do less work to group such states together. When one-hot encoding the ground states, for example, the Euclidean distance between the activations the encoded states $s_1 = [0, 1, 0.25]$ and $s_2 = [0, 1, 0.75]$, which belong to the same ground state $[0, 1]$, are mapped to will tend to already be smaller at the beginning of training than the one between the activations that the encoded states s_1 and $s_3 = [1, 0, 0.25]$, which belong to different ground states but have the same value for the superfluous feature, are mapped to. This form of superfluous feature encoding is denoted by (N).

3. The superfluous feature is randomly given one of 5 possible values between 0 and 4 and the superfluous feature is subsequently one-hot encoded. This encoding is solely utilized for one-hot encoded ground states and combines the advantages of the previous two forms of superfluous feature encoding when it comes to studying whether a DQN forms the coarsest Markov state representation. The reasons are that all features of the encoding have the same scale and that states corresponding to the same ground state are not already mapped closer together than states belonging to different ground states at the beginning of training. The states from the example above now would be encoded as $s_1 = [0, 1, 0, 1, 0, 0, 0], s_2 = [0, 1, 0, 0, 0, 1, 0]$ and $s_3 = [1, 0, 0, 1, 0, 0, 0]$. This means that the activations that s_1 and s_2 are mapped to do not tend to be closer in Euclidean distance than the ones s_1 and s_3 are mapped to at the beginning of training. We denote this type of superfluous feature encoding by (F-OH).

[14] Disparate results exist in the literature as to whether scaling of the input to the interval $[0, 1]$ or to $[-1, +1]$ is more effective for neural networks [10, 16]. The interval $[0, 1]$ is chosen for this work, because if the ground state is one-hot encoded, the resulting encoding still one-hot encodes the ground state and thus is more comparable to the other forms of state encoding that we experiment with.

In total, we employ 5 different types of state encoding for the Gridworld 3x3 and FrozenLake 4x4 domains and 3 ways of encoding states for the Gridworld 5x5 and FrozenLake 8x8 domains. Notice that the location-based encoding of ground states is used solely for the Gridworld 3x3 and FrozenLake 4x4 domains. The 5 types of state encoding are visualized in Figure 3.5.

3.1.4. DQN Implementation and Training

2- and 4-layer DQNs with hidden layers of sizes 1 to 50 are trained for each domain and for the previously described different types of state encoding. The correlation coefficients are thereby computed for each training episode. In addition, we test the networks and compute the number of optimal actions learned after each training episode. Based on these experiments, we use the following terminology when referring to different hidden layer sizes for a network:

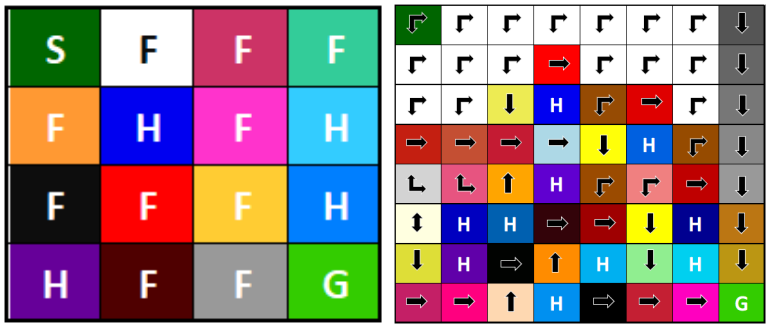
- *Just-right hidden layer size*. This is the smallest hidden layer size for which a network converges to the optimal policy at least one time out of 5 times in our experiments.
- *Larger-than-necessary hidden layer sizes*. These are all hidden layer sizes larger than the smallest hidden layer size for which a network converges to the optimal policy at least one out of 5 times in our experiments.
- *Sufficiently large hidden layer sizes*. These are all hidden layer sizes that allow a DQN to converge to the optimal policy at least one out of 5 times in our experiments.

Furthermore, we say that the test rewards have converged when an agent arrives at a goal state for each test episode at the end of training. The detailed training procedure and network architectures are described in Section D.2 in the Appendix.

3.1.5. t-SNE Visualization

To visualize internal state representations, we create t-SNE plots as explained in Section 2.8 with a default perplexity value of 30[15]. Thereby, the activations of all states belonging to the same ground state are drawn in the same color. Specifically, this means that the activations of all states that differ solely in the superfluous feature value are depicted in the same color.

FrozenLake. For the FrozenLake domains, states corresponding to ground states with similar best and worst actions are further depicted in similar colors. For example, all ground states with the optimal action ”down” and the worst action ”right” in the FrozenLake 8x8 domain are drawn in a shade of yellow (see Figure 3.6b). In addition, all states corresponding to holes in the FrozenLake domains are visualized in a shade of blue or purple and those corresponding to the start and goal state in dark and bright green, respectively. Notice that all terminal states both are bisimilar and have the same Q-values. The precise coloring schemes are shown in Figure 3.6.

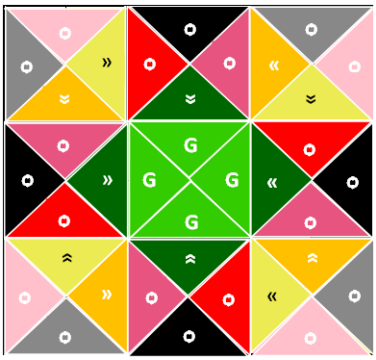


(a) FrozenLake 4x4. (b) FrozenLake 8x8.

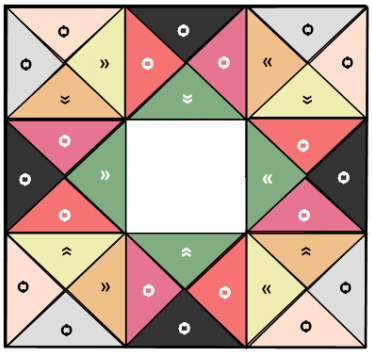
Figure 3.6: Colors used for each ground state of the FrozenLake domains in t-SNE plots. All states corresponding to holes are depicted in a shade of blue or purple, the start state in dark green and the goal state in bright green. All other states are colored so that states with similar best and worst actions have similar colors. Note that the ground states colored in white for FrozenLake 8x8 are not bisimilar.

[15] Section C.6 in the Appendix exemplarily shows the impact of different perplexity values on the t-SNE plot of the activations states are mapped to in the hidden layer of a 2-layer DQN and justifies our setting of 30.

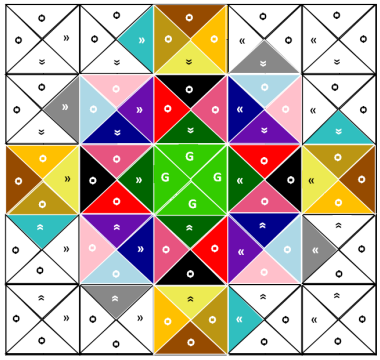
Gridworld. For the Gridworld domains, states corresponding to ground states with the same sequence of optimal actions until reaching a goal state, which both are bisimilar and have the same Q-values, are further shown in the same color. For instance, all ground states from which the agent most quickly reaches a goal state via the actions $\langle rotate, rotate, rotate, forward \rangle$ are visualized in red for the Gridworld 3x3 domain (see Figure 3.7a). If too many different colors would be required to distinguish all groups of ground states with the same future optimal action sequences, only a selected number of such groups is highlighted and the remaining ones are visualized in white. For the augmented Gridworld 3x3 domain, all states corresponding to copied ground states are further depicted in slightly paler colors than states belonging to the corresponding original ground states as shown in Figure 3.7b. Moreover, states corresponding to the additional terminal state are drawn in white. The exact coloring schemes are depicted in Figure 3.7.



(a) Gridworld 3x3.



(b) Gridworld 3x3 (Aug) - Copied states.



(c) Gridworld 5x5.

Figure 3.7: Colors used for each ground state of the Gridworld domains in the t-SNE plots. For each grid location, four triangles represent the possible orientations of the agent. A circle denotes that the optimal action in a state is $rotate$ and arrows that the optimal action is $forward$. All ground states drawn in the same color are bisimilar unless they are drawn in white.

3.2. Results

To answer the question of what a DQN learns during training, we identify and describe three learning phases for the hidden- and output-layer representations of DQNs in Section 3.2.1. Furthermore, we examine several factors that impact the internal state representations formed during these three phases in Section 3.2.2. We for sake of simplicity look at 2-layer DQNs by default, yet also explore the impact of using three rather than a single hidden layer.

3.2.1. Learning Process

Figure 3.8 exemplarily shows for a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain that the learning process of DQNs consists of three overlapping learning phases for the hidden and output layers:

1. *States are grouped based on multi-step rewards*. This process is visualized by the initial steep increase and subsequent step-wise decrease in the reward-based correlation coefficient c_{Rew}.
2. *The internal state representations become more similar to the coarsest Markov state representation and more Markov with respect to the transition function*. This pattern is mirrored by the increase in the correlation coefficients based on d_{fix} and T_{TV}, $c_{K(d_{fix})}$ and c_{TV}, at the beginning of training.
3. *States are increasingly clustered based on Q-values*, as visualized by the step-wise increase in the Q^*-based correlation coefficient, c_{Q^*}, after an initial plateau.

In the following, we will discuss each of these three phases using a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain as primary example.

3.2.1.1 Learning Phase 1

Since the target network provides the estimates of the Q-values of next states during training, it is not surprising that the activations of states with the same $n+1$-step rewards tend to be grouped together,

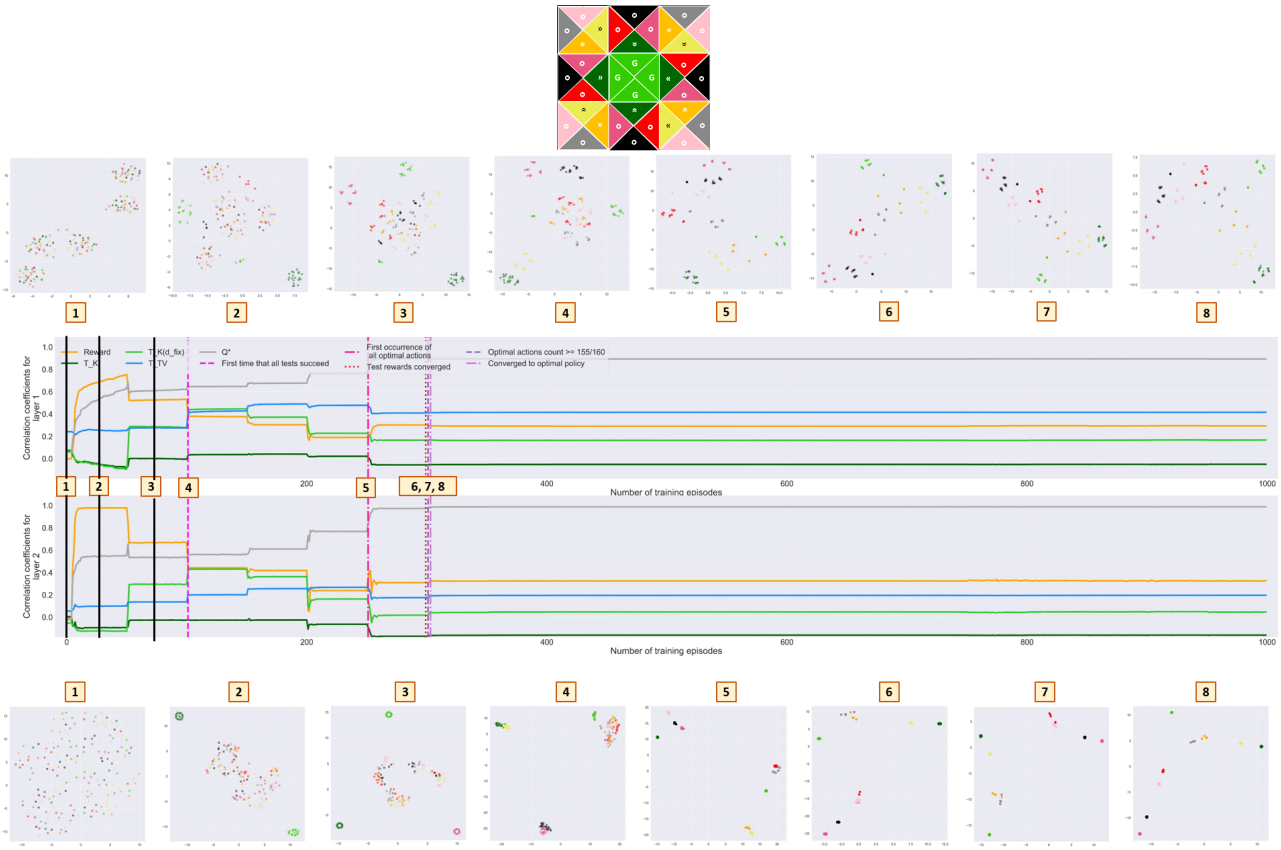


Figure 3.8: Correlation coefficients and t-SNE plots of the activations the states are mapped to during training for the layers of a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain. The hidden layer size is equal to 50 and the target network is updated every 50 training episodes. The t-SNE coloring scheme is shown at the top.

where n is the number of times the target network has been updated. For example, Figure 3.8 shows that a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain forms a separate cluster for the states whose activations are colored in dark green while the target network has not yet been updated. This is the case, because those states have an immediate reward of 1 for one action whereas all other states have an immediate reward of 0 for all actions. Once the target network has been updated once, separate clusters are also created for the states whose activations are drawn in dark pink, because those states have a non-zero two-step reward for one action. This pattern of grouping states based on $n+1$-step rewards can nicely be measured by the reward-based correlation coefficient c_{Rew}. The reason is that c_{Rew} takes on the highest value if states are distinguished solely based on immediate rewards and progressively lower values when more distinctions are made. This behavior is visualized in Figure 3.8, where c_{Rew} quickly increases to a value near 1 at the beginning of training and subsequently decreases for each of the first 4 times that the target network is updated[16].

3.2.1.2 Learning Phase 2

Figure 3.8 depicts that the formed first- and second-layer representations become progressively more Markov with respect to the transition function and closer to the coarsest Markov state representation, indicated by c_{TV} and $c_{K(d_{fix})}$ respectively, at the beginning of training. This intuitively makes sense, because as DQNs learn to output multi-step rewards, they also need to represent the transition function in some way. For instance, t-SNE plots 2 in Figure 3.8 visualize that before the target network is updated for the first time, most terminal states, which are drawn in bright green, already form a distinct cluster. This occurs despite the fact that the immediate rewards for all actions are equal to 0 for the terminal states just like for all other states whose activations are not shown in dark green. Hence, the DQN has

[16] In the Gridworld 3x3 domain, at most 5 steps are needed to reach a goal state from any state under π^*. So when the target network has been updated 4 times, the targets during training are based on 5 steps and thus significantly different from 0 for each state's optimal action. Once the target network is updated one more time, the DQN converges to the true Q-values, as Q-values are based on at most 6 transitions for this domain.

not only learned to represent the immediate rewards for the states whose activations are colored in dark green, but also the transition function.

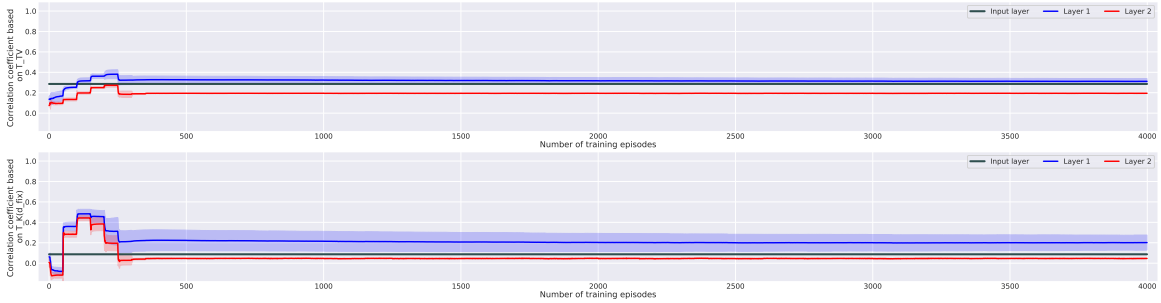


Figure 3.9: Mean c_{TV} and $c_{K(d_{fix})}$ for the layers of 2-layer DQNs with a hidden layer size of 8 for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions and 95%-confidence intervals are shown. The term *input layer* refers to the state encoding.

The first-layer representation tends to be more Markov with respect to the transition function and closer to the coarsest Markov state representation than both the encoding of the input and the output-layer representation. It is therefore not the case that the first layer just forms a representation that falls between the encoding of the input and the representation learned by the output layer. This is visualized exemplarily in Figure 3.9 for a 2-layer DQN with a hidden layer size of 8 for the Gridworld 3x3 (OH)(F-OH) domain, where c_{TV} and $c_{K(d_{fix})}$ are typically higher in the first than in the input[17] and output layers[18]. The higher values of c_{TV} and $c_{K(d_{fix})}$ in the first network layer compared to the output layer largely stem from different distances between the abstract states that are created. For instance, Figure 3.10 shows that once the target network has been updated twice for the Gridworld 3x3 (OH)(F-OH) domain, the same clusters of states are present in the first and the output layer. Due to the higher value for $c_{K(d_{fix})}$ in the first than in the second layer, the inter-cluster distances in the first layer are thus based more on bisimilarity and less on multi-step rewards than in the output layer[19]. Furthermore, the higher value for c_{TV} in the first layer additionally results from slightly less similar activations for bisimilar states from different ground states in the first than in the output layer[20]. For example, the dark pink and black clusters in Figure 3.10a are made up of 4 sub-clusters, one for each ground state.

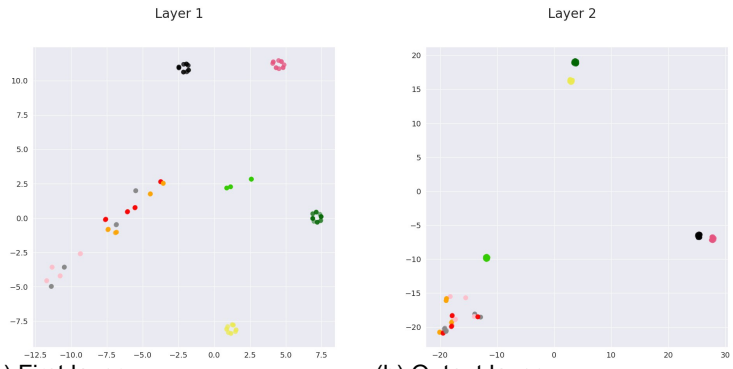
Supporting evidence. We provide experimental results for three more domains in Section A.1.1.1. Moreover, one might argue that the observation that the internal state representations become more similar to the coarsest Markov state representation during this phase of training is solely the byproduct of clustering states based on multi-step rewards or the result of the way a domain is explored. We thus conducted further experiments, which show that the hidden- and output-layer state representations also become more similar to the coarsest Markov state representation during this phase of learning in a domain in which states with the same Q-values are not necessarily bisimilar, and when a fixed replay memory is used during training in Sections A.1.1.2 and A.1.1.3. Furthermore, the first-layer representation tends to also be more similar to the coarsest Markov state representation and more Markov with respect to the transition function than the ones in subsequent hidden layers as discussed

[17] The term *input layer* refers to the state encoding. There are hence no weights to tune for this layer.

[18] In addition, Figure A.24 in the Appendix makes clear for the Gridworld 3x3 (OH)(F-OH) domain that the peak first-layer values of c_{TV} and $c_{K(d_{fix})}$ are on average higher than the ones in the output layer and the constant input-layer values for all just-right and larger-than-necessary hidden layer sizes. Note that looking at the mean peak values of c_{TV} and $c_{K(d_{fix})}$ is useful for a Gridworld domain, because the peaks occur for both layers at similar times during the beginning of training and c_{TV} and $c_{K(d_{fix})}$ subsequently decrease (see Figure 3.9).

[19] One could be tempted to draw this conclusion based on the inter-cluster distances in the t-SNE plots in Figure 3.10 alone. For instance, the activations drawn in black and dark pink are very similar compared to the other activations in the output but not the first layer. While the states corresponding to black and dark pink activations have very similar two-step rewards compared to the other states, they do not have relatively low distances with respect to d_{fix} or T_{TV} compared to the other states. Yet, the distances between well separated clusters in t-SNE plots are not necessarily meaningful as discussed in Section 2.8.

[20] Recall that c_{TV} measures how well the *next state of the input MDP* can be predicted and not necessarily how well *next abstract states* under the notion of bisimulation can be predicted.



(a) First layer. (b) Output layer.

Figure 3.10: t-SNE plots of the activations states are mapped to in the layers of a 2-layer DQN with a hidden layer size of 3 for the Gridworld 3x3 (OH)(F-OH) domain after training episode 125. At this point during training, the target network has been updated twice.

in Section A.1.1.4. Lastly, Section A.1.1.5 demonstrates that both $c_{K(d_{fix})}$ and $c_{d'_{fix}}$, the latter of which is much faster to compute, enable insights into how similar to the coarsest Markov state representation an internal state representation is.

3.2.1.3 Learning Phase 3

While states are increasingly clustered based on bisimilarity in the network layers during the second learning phase, states are at some point progressively grouped by Q-values rather than bisimilarity. Generally, forming a Q^*-irrelevance abstraction leads to an increase in c_{Q^*}. This is depicted in Figure 3.8, where c_{Q^*} begins to increase again near training episode 150 after largely plateauing beforehand. Ultimately, c_{Q^*} reaches a value near 1 in both network layers when the DQN converges to the optimal policy[21]. At the same time, the value for $c_{K(d_{fix})}$ decreases for this domain as the inter-cluster distances become more and more different from those of the coarsest Markov state representation[22]. This is visualized in Figure 3.8, where $c_{K(d_{fix})}$ begins to decrease exactly when c_{Q^*} increases again. The final internal state representations present at the end of training are therefore less similar to the coarsest Markov state representation for this domain than they were during the second phase of learning.

3.2.2. Factors Impacting the Internal State Representations

Several factors impact which internal state representations are formed in the layers of DQNs. We discuss factors that influence the representations in all layers during the first and third learning phases in Section 3.2.2.1 and Section 3.2.2.2, respectively, and aspects impacting the hidden-layer state representation during the last two learning phases in Section 3.2.2.3.

3.2.2.1 Internal State Representations During Learning Phase 1

Whether and how quickly effective hidden- and output-layer representations are created during the first learning phase is contingent on the available data and the hidden layer size.

Available data. While the same pattern of grouping states based on $n+1$-step rewards is visible for a 2-layer DQN for the Gridworld 5x5 (OH)(F-OH) domain as depicted in Figure A.1 in the Appendix, the process of learning is slightly different for the FrozenLake domains. Figure 3.11 shows that it takes longer to fully group states based on immediate rewards at the beginning of training for the FrozenLake 8x8 (OH)(F-OH) domain. The reason for this is the existence of non-goal terminal states, which make it more difficult to explore states farther away from the fixed start state. Therefore, only states with the same immediate rewards that are relatively close to the start state are initially grouped together. Yet, once sufficient data on states closer to the goal state is available, the activations of all states are also increasingly clustered based on $n+1$-step rewards. This is visualized by the step-like decrease in the

[21] However, c_{Q^*} is not exactly equal to 1 for a Q^*-irrelevance abstraction, as it only captures the maximum absolute distance in Q-values for any action and hence not the differences in Q-values for all actions.

[22] Note that as states are bisimilar if and only if they have the same Q-values in the Gridworld 3x3 domain, the clusters themselves of the coarsest Markov state representation and of a Q^*-irrelevance abstraction are equal.

reward-based correlation coefficient, c_{Rew}, after an initial peak in Figure 3.11[23]. Consequently, as a network can only effectively cluster states based on multi-step rewards when data on many states is available, a good exploration strategy at the beginning of training is important.

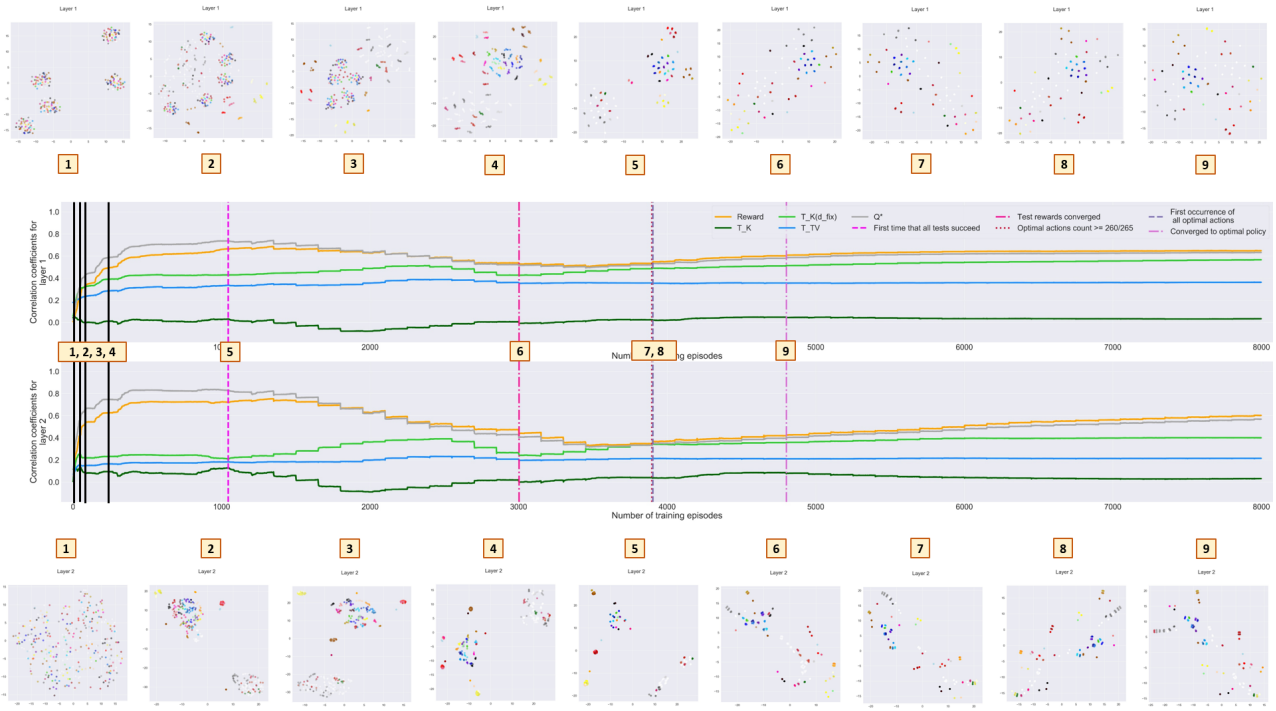


Figure 3.11: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the FrozenLake 8x8 (OH)(F-OH) domain. The hidden layer size is equal to 50.

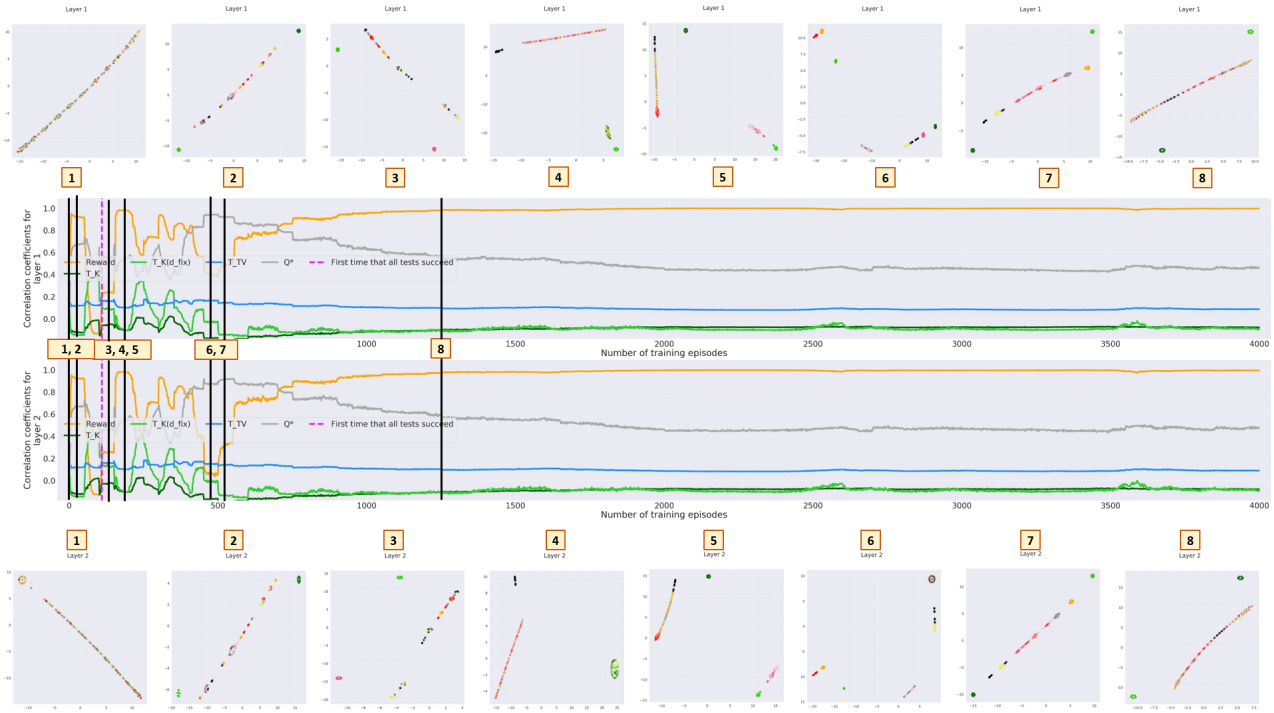


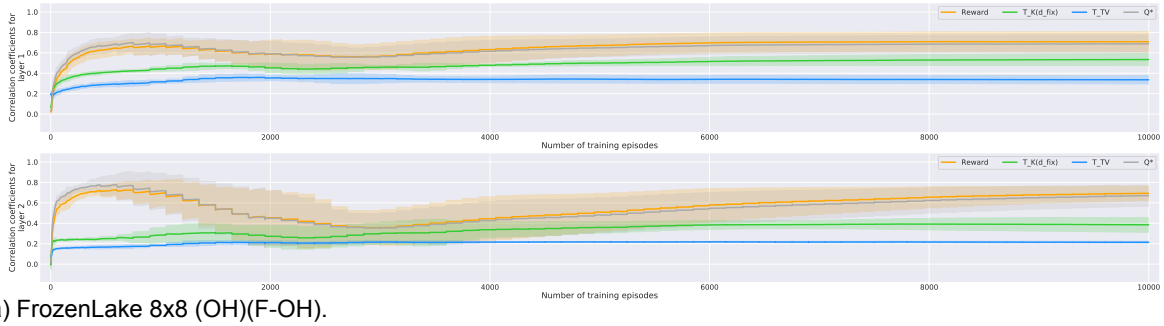
Figure 3.12: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the Gridworld 3x3 (OH) domain. The hidden layer size is equal to 1.

[23] In the FrozenLake 8x8 domain, Q-values are based on at most 15 transitions, which roughly corresponds to the number of target network updates for which c_{Rew} decreases after its initial peak in Figure 3.11.

Hidden layer size. If the hidden layer size is very small, states may solely be grouped based on rewards that take a limited number of steps into consideration. For instance, Figure 3.12 shows that a 2-layer DQN with a hidden layer size of 1 for the Gridworld 3x3 (OH) domain does not succeed at fully clustering states based on more than two-step rewards in its two layers. Instead, after grouping states based on one- and two-step rewards and partially based on three-step rewards, states are largely clustered based on immediate rewards again. The latter is indicated by the separate cluster for the states whose activations are depicted in dark green in t-SNE plot 5 in Figure 3.12 and the accompanying high value of c_{Rew}. As will be discussed in Section 3.2.2.3, a hidden layer size of 1 makes it impossible to form close to the coarsest Markov state representation or close to a Q^*-irrelevance abstraction in the first layer, which explains this phenomenon. Hence, to enable clustering of states based on multi-step rewards, it is important to choose a sufficiently large hidden layer size.

3.2.2.2 Internal State Representations During Learning Phase 3

The extent to which the internal state representations in hidden and output layers are similar to the coarsest Markov state representation during learning phase 3 and at the end of training depends on domain characteristics. For example, even though states are bisimilar if and only if they have the same Q-values in both FrozenLake 8x8 and Gridworld 3x3, the internal state representations become progressively more similar to the coarsest Markov state representation during the third learning phase for FrozenLake 8x8, but increasingly less similar for the Gridworld 3x3 domain. This is visualized in Figure 3.13, where $c_{K(d_{fix})}$ increases towards the end of training in the layers of a 2-layer DQN for the FrozenLake 8x8 domain, but ultimately decreases in the layers of a 2-layer DQN for the Gridworld 3x3 domain. This is the case, because the inter-cluster distances of a Q^*-irrelevance abstraction and the coarsest Markov state representation are more similar for FrozenLake 8x8 than for Gridworld 3x3.



(a) FrozenLake 8x8 (OH)(F-OH).

(b) Gridworld 3x3 (OH)(F-OH).

Figure 3.13: Mean c_{Rew}, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in each layer with 95%-confidence intervals for a 2-layer DQN for different domains. Values are based on 5 repetitions. The hidden layer size is equal to 50.

Besides $c_{K(d_{fix})}$, the values of the other correlation coefficients for a Q^*-irrelevance abstraction also are domain-dependent. This is mirrored by the mean values of the correlation coefficients in the output layers at the end of training in Figure 3.13, because the output layer of a DQN ultimately creates a Q^*-irrelevance abstraction if possible. Hence, based on the values of the bisimulation-based correlation coefficients alone, one cannot determine the degree to which a Q^*-irrelevance abstraction has been formed. Moreover, computing the bisimulation-based correlation coefficients does not allow drawing conclusions as to whether a DQN has discovered or converged to the optimal policy as discussed in Section A.1.3 in the Appendix.

3.2.2.3 Hidden-layer State Representations During Learning Phases 2 and 3

The discussion of the three learning phases shows that learning phases 2 and 3 are the phases during which the degrees to which the internal state representations are similar to the coarsest Markov state representation change. Thereby, especially the representations in hidden layers are of interest, since the output layer of a DQN is constrained to learn to represent the Q-values. We find that several factors impact what kind of internal state representation a DQN learns in its hidden layers during these phases. More precisely, the representations in hidden layers become closer to the coarsest Markov state representation, and are subsequently further abstracted towards a Q^*-irrelevance abstraction, to an extent that depends on the necessity, difficulty and feasibility of forming such representations. Among the factors impacting the necessity, difficulty and feasibility are the network capacity as expressed by the hidden layer size and the number of hidden layers, the state space size and the state encoding. In the following, we will use the first layer of a 2-layer DQN for the Gridworld 3x3 domain as primary example to illustrate our claims. Supporting evidence based on the FrozenLake 4x4, FrozenLake 8x8 and Gridworld 5x5 domains is given in Section A.1.2 in the Appendix.

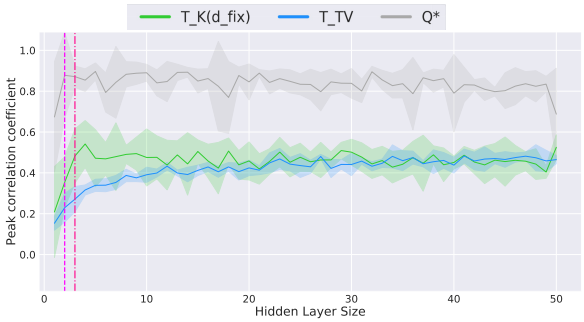


Figure 3.14: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

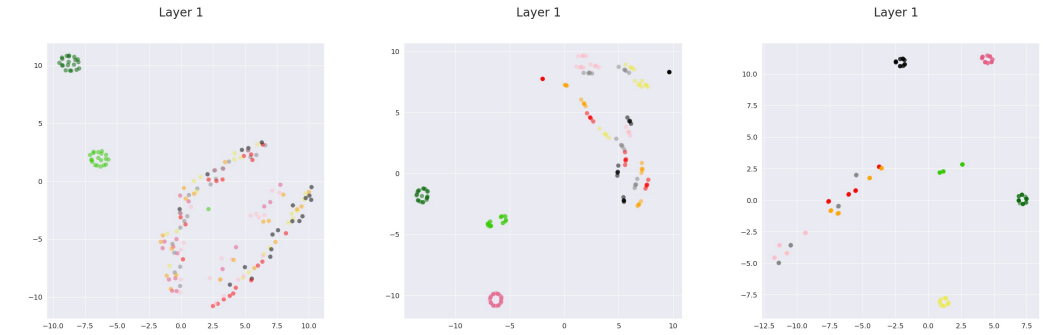
Necessity. The extent to which the first-layer representation becomes similar to the coarsest Markov state representation during training and is subsequently further abstracted towards a Q^*-irrelevance abstraction depends on the degree to which this is required for the network to be able to learn the true Q-values, which is contingent on the network capacity[24]. Both the hidden layer size and the number of hidden layers influence the capacity of a network, whereby a network's flexibility is higher for larger and more hidden layers. Below, we discuss the impact of the hidden layer size and the number of hidden layers on the formed first-layer state representation.

- **Similarity to a Q^*-irrelevance abstraction**. The lower the hidden layer size, the more similar to a Q^*-irrelevance abstraction of the state space does the first layer need to form during the third learning phase to allow the network to converge to the true Q-values. This becomes evident in Figure 3.14, which depicts the peak values of the correlation coefficients for the Gridworld 3x3 (OH)(F-OH) domain for various hidden layer sizes. Recall that the peak value for c_{Q^*} is attained at the end of training for the Gridworld domains. Thus, these figures allow us to conclude that c_{Q^*} takes on the highest value in the first layer at the end of training for hidden layer sizes that are just large enough to enable the DQN to converge to the optimal policy, and progressively lower values for larger hidden layer sizes. A similar observation can be made for the Gridworld 5x5 domain and the FrozenLake domains in Figures A.9 and A.10[25], respectively, in the Appendix.

- **Similarity to the coarsest Markov state representation**. The first-layer representation also tends to become more similar to the coarsest Markov state representation during the second learning phase when the network's flexibility is lower. For example, Figure 3.14 shows that the

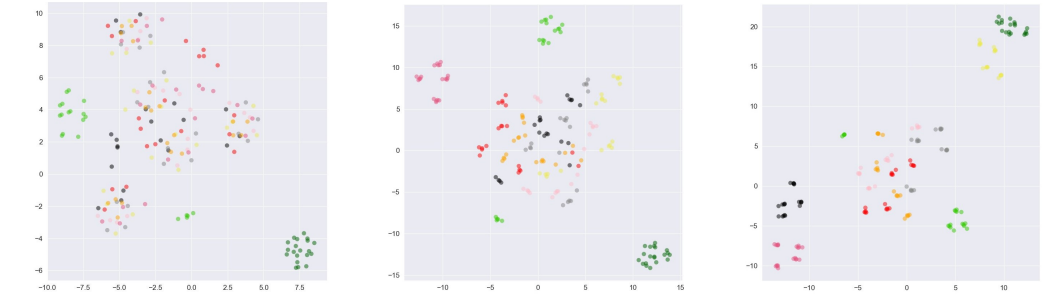
[24] While one could thus assume that computing the correlation coefficients allows drawing conclusions regarding the adequacy of a network's capacity, Section A.1.4 in the Appendix demonstrates that this is not the case.

[25] Notice that this figure shows the final values of the correlation coefficients at the end of training instead of the peak values, because the peak value for c_{Q^*} for the FrozenLake 8x8 domain does not necessarily occur at the end of training.



Training episode 25. Training episode 75. Training episode 125.

(a) $hidden\_layer\_size = 3$.

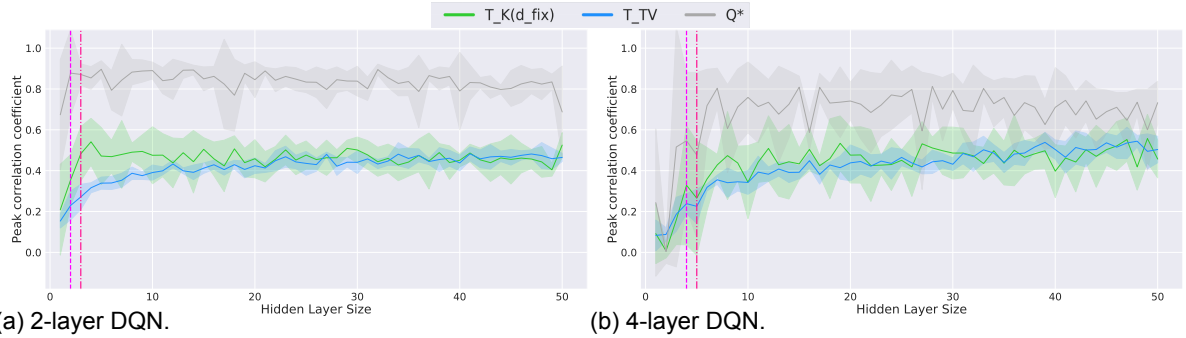


Training episode 25. Training episode 75. Training episode 125.

(b) $hidden\_layer\_size = 50$.

Figure 3.15: t-SNE plots of the activations the states are mapped to for the first layers of 2-layer DQNs with different hidden layer sizes for the Gridworld 3x3 (OH)(F-OH) domain. Since the target network is updated every 50 episodes for this domain, it has been updated 0, 1 and 2 times at training episodes 25, 75 and 125, respectively.

mean peak value of $c_{K(d_{fix})}$ in the first layer of a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain is lower for much larger-than-necessary hidden layer sizes than for just-right and slightly larger ones. As depicted in Figure 3.15 for 2-layer DQNs for the Gridworld 3x3 (OH)(F-OH) domain, the decreasing extent to which the first-layer representation becomes similar to the coarsest Markov state representation for larger hidden layer sizes is due to both bisimilar states from different ground states and bisimilar states differing solely in the superfluous feature value being mapped to less similar activations[26]. Similarly, the first-layer representation of a 2-layer DQN becomes close to the coarsest Markov state representation to higher degrees than the one of a 4-layer DQN for just-right hidden layer sizes (see Figure 3.16).



(a) 2-layer DQN. (b) 4-layer DQN.

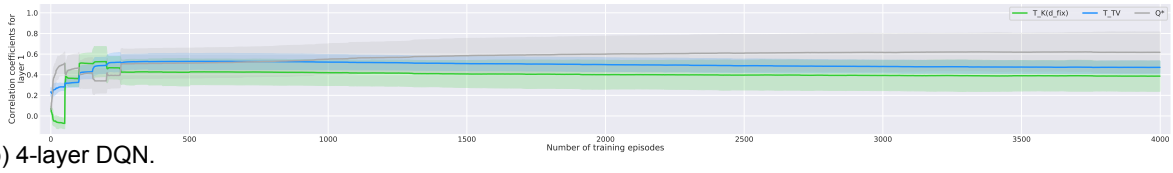
Figure 3.16: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2- and a 4-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

Yet, while increasing the network capacity generally causes the first-layer representation to be-

[26] Recall that always 4 ground states are bisimilar in the Gridworld 3x3 domain. There tend to be single clusters for the states corresponding to such 4 ground states in Figure 3.15a. The t-SNE plots in Figure 3.15b, however, typically contain clearly distinct clusters for each ground state or even for states belonging to the same ground state.

come *less* similar to the coarsest Markov state representation, using network capacities slightly larger than necessary can lead to the first-layer representation becoming *more* similar to the coarsest Markov state representation. This is, for example, the case for larger-than-necessary hidden layer sizes when employing three rather than a single hidden layer. The reason is that the first-layer representation needs to be abstracted towards a Q^*-irrelevance abstraction to a lesser extent for 4- than for 2-layer DQNs, and the first layer of a 4-layer DQN can thus form a representation that is more similar to the coarsest Markov state representation (see Figure 3.17). Thus, using slightly larger-than-necessary network capacities may allow the first-layer representation to become more similar to the coarsest Markov state representation.

(a) 2-layer DQN.

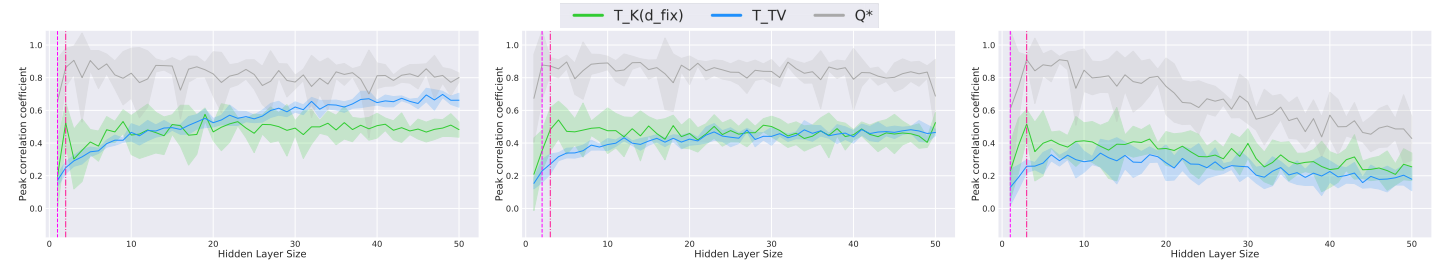


(b) 4-layer DQN.

Figure 3.17: Mean $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for the first layers of 2- and 4-layer DQNs with a hidden layer size of 47 for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions and 95%-confidence intervals are shown.

- **Variability**. A higher network capacity additionally leads to more variability in the learned first-layer representation. The reason is that aspects such as the initialization of the weights and the order in which transitions are sampled from the replay memory during training have a stronger impact on the first-layer representation when the network capacity is larger. This becomes evident when comparing the standard deviations of the correlation coefficients for 2- and 4-layer DQNs in Figures 3.16 and 3.17.

Difficulty. For larger-than-necessary hidden layer sizes, the difficulty of the problem also impacts which first-layer state representations are created. In our experiments, we varied the state encoding and used domains with different state space sizes. Thereby, we discovered that when the state encoding or the state space size render it more difficult to form a first-layer state representation that is similar to the coarsest Markov state representation and such a representation is not required due to a network's larger-than-necessary capacity, such a representation is learned to a lesser extent. Similar observations hold for a Q^*-irrelevance abstraction. We discuss the impact of the state encoding and the state space size below.

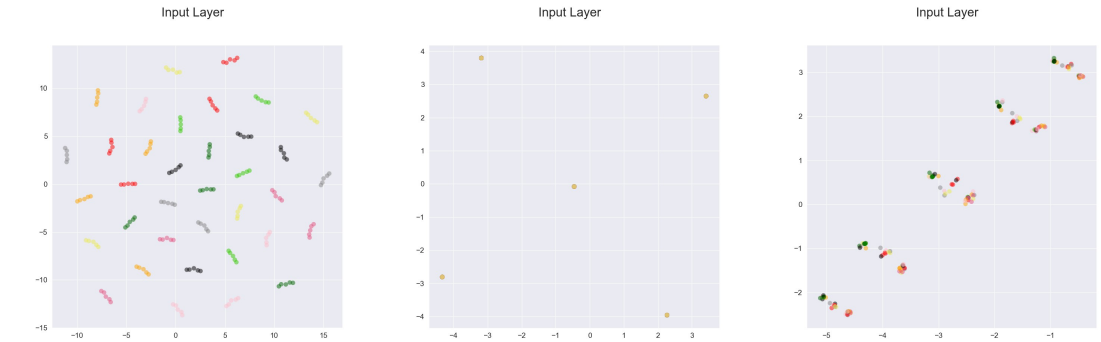


(a) Gridworld 3x3 (OH)(N). (b) Gridworld 3x3 (OH)(F-OH). (c) Gridworld 3x3 (OH).

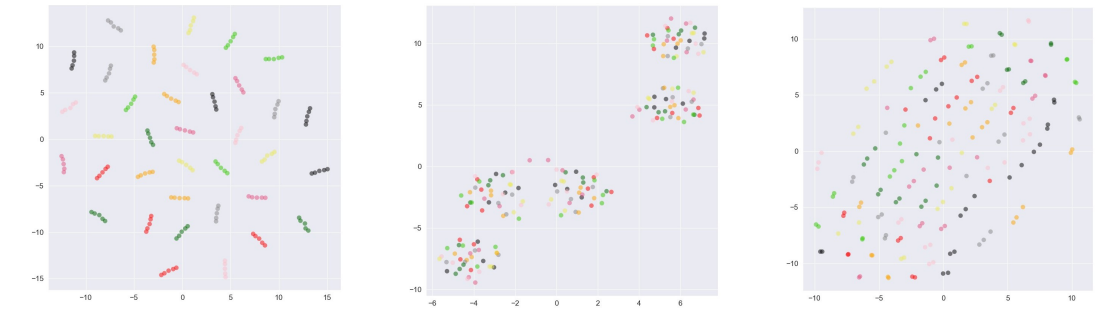
Figure 3.18: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the Gridworld 3x3 domain with different forms of state encoding. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

- **State encoding**. The first-layer representation becomes most similar to the coarsest Markov state representation and ultimately to a Q^*-irrelevance abstraction for larger-than-necessary hidden layer sizes for the (OH)(N) state encoding, followed by the (OH)(F-OH) encoding and finally

by the (OH) encoding. This can be seen from the mean peak values for $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in Figure 3.18[27]. As visualized in Figure 3.19, this can be attributed to the first-layer representations at the beginning of training as well as the input-layer representations[28]. While states differing solely in the superfluous feature value are already mapped to very similar activations in the initial first-layer and the input-layer representation for the (OH)(N) encoding, the activations of states corresponding to the same ground state have relatively large Euclidean distances in the initial first-layer and the input-layer representation for the (OH) encoding. Moreover, the Euclidean distances between the input-layer and initial first-layer activations of states differing only in the superfluous feature value for the (OH)(F-OH) encoding fall somewhere in between the ones for the other two types of state encoding[29]. Hence, it is most difficult to map bisimilar states differing solely in the superfluous feature value, which also have the same Q-values, to similar activations for the (OH) state encoding and easiest for the (OH)(N) encoding[30].



Gridworld 3x3 (OH)(N). Gridworld 3x3 (OH)(F-OH). Gridworld 3x3 (OH).
(a) Input Layer.



Gridworld 3x3 (OH)(N). Gridworld 3x3 (OH)(F-OH). Gridworld 3x3 (OH).
(b) First Layer.

Figure 3.19: t-SNE plots of the activations at the beginning of training for the input and first layers of 2-layer DQNs with a hidden layer size of 50 for the Gridworld 3x3 domain with different forms of state encoding. Activations are computed for all states and every 5 states that differ solely in the superfluous feature and hence correspond to the same ground state are drawn in the same color. In addition, states corresponding to bisimilar ground states are shown in the same color.

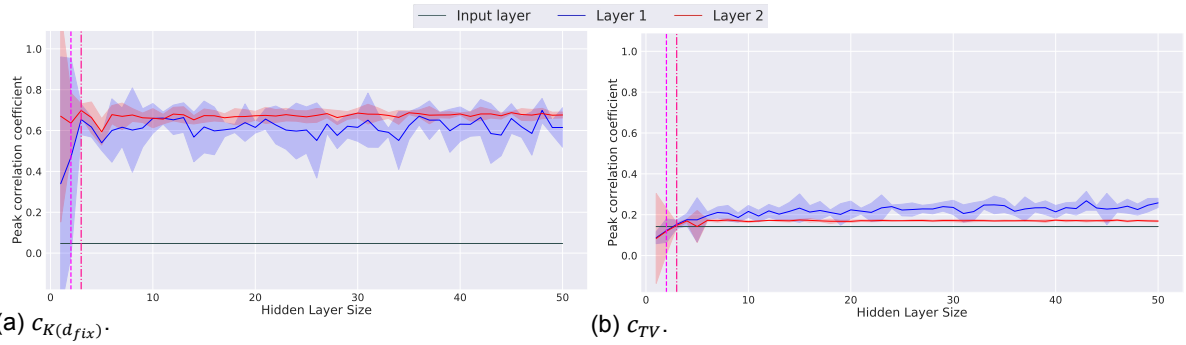
- **State space size**. The first-layer representation becomes relatively less similar to the coarsest Markov state representation and Markov with respect to the transition function for domains with

[27] The peak value for c_{TV} does not always arise during the second learning phase for the (OH)(N) encoding. Instead, the peak value for c_{TV} is the initial value for hidden layer sizes larger than or equal to the dimensionality of the state encoding (see Figures A.25b and A.26b in the Appendix). Thus, while the peak value for c_{TV} for the (OH)(N) encoding does not show how Markov with respect to the transition function the first-layer representation becomes during the second learning phase, it clearly expresses how much easier it is for this type of state encoding to group states from the same ground states together in the first layer due to the favorable initialization.

[28] As the weights of the linear layers of our networks are randomly initialized with small values near 0, the state encoding strongly impacts the state representation in the first network layer at the beginning of training. However, if the size of the first layer is much smaller than the one of the input layer, the correlation coefficients in the first layer at the beginning of training tend to be more different from those in the input layer due to the higher degree of compression.

[29] Notice that even though there are five clusters in the t-SNE plot for the input-layer representation for the (OH)(F-OH) encoding just as for the (OH) encoding, these clusters are equidistant and have a Euclidean distance of 1. For the (OH) encoding, however, the five clusters in the input layer are not equidistant and the Euclidean distance between states differing solely in the superfluous feature value can be as large as 2. Refer to Section 3.1.3 for more information on the types of state encoding.

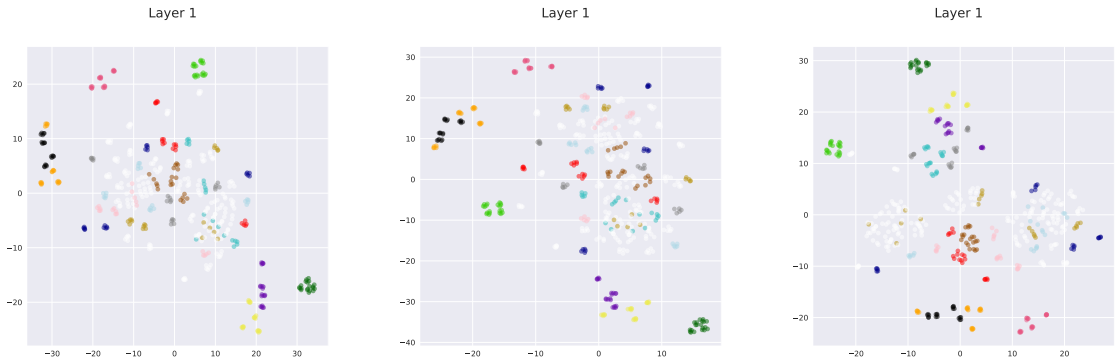
[30] Similar observations can be made for the FrozenLake 4x4 domain in Figures A.11 and A.12 in the Appendix.



(a) $c_{K(d_{fix})}$. (b) c_{TV}.

Figure 3.20: Mean peak $c_{K(d_{fix})}$ and c_{TV} in each layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the Gridworld 5x5 (OH)(F-OH) domain. Values are based on 4 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 4 times.

larger state spaces. For instance, Figure 3.20 reveals that even for just-right hidden layer sizes, the peak values of $c_{K(d_{fix})}$ and c_{TV} are not significantly higher in the first than in the output layers of DQNs for Gridworld 5x5, whereas this clearly is the case for Gridworld 3x3[31]. This is caused by two aspects. First, the Gridworld 3x3 and 5x5 domains have very similar just-right hidden layer sizes when the ground state is one-hot encoded. Due to the larger state space size of the Gridworld 5x5 domain, those just-right hidden layer sizes necessitate a stronger compression of the state space for the Gridworld 5x5 than the Gridworld 3x3 domain. This means that the first-layer representation is further abstracted towards a Q^*-irrelevance abstraction more quickly and more intensely for the Gridworld 5x5 domain. Second, mapping states with the same multi-step rewards from different ground states to very similar activations in the first network layer takes longer for larger state spaces, because more weights have to be learned. Hence, for larger-than-necessary hidden layer sizes for such domains, the second layer learns to represent the Q-values based on a certain number of transitions before the first-layer representation has fully created the corresponding clusters. For example, in the t-SNE plot for the first-layer representation in Figure 3.21c, the activations drawn in purple, light blue and gray, which correspond to non-bisimilar states with similar Q-values, fall into a single cluster. Yet, the states whose activations are colored in light blue or gray have not formed distinct clusters beforehand (see Figures 3.21a and 3.21b).



(a) Training episode 300. (b) Training episode 301. (c) Training episode 302.

Figure 3.21: t-SNE plot of the activations in the first layer of a 2-layer DQN for the Gridworld 5x5 (OH)(F-OH) domain for different training episodes. The hidden layer size is 50. Refer to Figure 3.7c for the coloring scheme.

Feasibility. The state encoding and hidden layer size not only impact how difficult and necessary it is to form a first-layer representation close to the coarsest Markov state representation and subsequently one similar to a Q^*-irrelevance abstraction, but also whether it is possible in the first place. It is thus crucial to select an appropriate form of state encoding and sufficiently large hidden layers.

[31] See Figure A.24 for the corresponding plots for the Gridworld 3x3 domain.

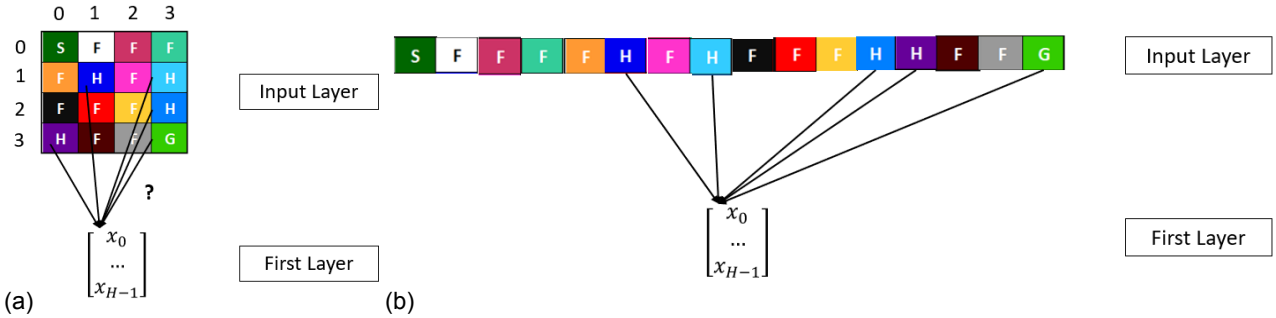


Figure 3.22: To obtain the coarsest Markov state representation for the FrozenLake 4x4 domain, all terminal states need to be mapped to the same activation pattern and all other ground states to distinct activations. a) When the ground state is encoded via features, the ground states are encoded based on their x- and y-coordinates in the grid and mapping states to the same first-layer activations if and only if they are bisimilar requires non-linearly separable functions to be learned. b) When the ground state is one-hot encoded, a linearly separable function is sufficient to group states together if and only if they are bisimilar.

- **State encoding**. Grouping states together in the first layer if and only if they are bisimilar or if and only if they have the same Q-values is not possible for the (F) and (F)(N) forms of state encoding for the Gridworld 3x3 and FrozenLake 4x4 domains[32]. This is the case, because a single-layer neural network can only learn linearly separable functions for each of its output nodes [53]. Yet, bisimilar states are not linearly separable for these domains when the ground state is encoded via features (see Figure 3.22a). Notice that when the ground state is one-hot encoded, mapping states to the same activations if and only if they are bisimilar requires solely a linearly separable function to be learned.

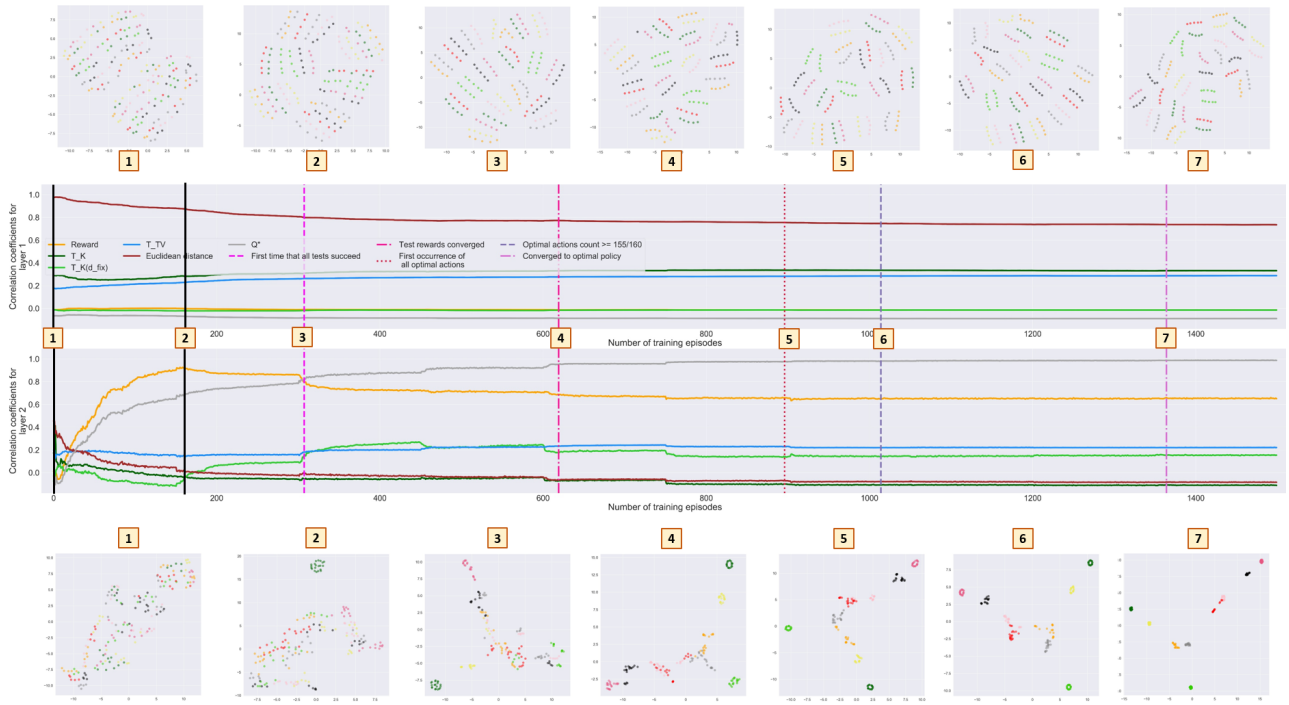
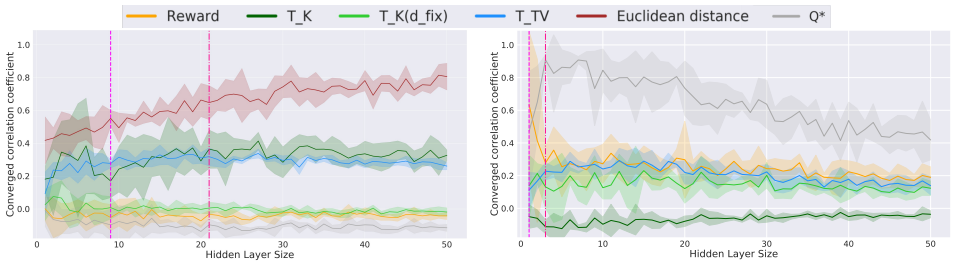


Figure 3.23: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (F) domain.

Rather than grouping states based on Q-values or bisimilarity for these types of state encoding, the first-layer representation learns to cluster states based on their ground states as shown in

[32] Recall that states are bisimilar if and only if they have the same Q-values in the Gridworld 3x3, Gridworld 5x5, FrozenLake 4x4 and FrozenLake 8x8 domains.

Figure 3.23 for the Gridworld 3x3 (F) domain[33][34]. Since the learned first-layer representation consequently is a finer abstraction of the state space than both the coarsest Markov state representation and a Q^*-irrelevance abstraction, more hidden nodes are required for 2-layer DQNs for the Gridworld 3x3 (F) domain to converge to the optimal policy. This is indicated by the vertical lines in Figure 3.24. More precisely, whereas fewer than 5 hidden nodes are needed for 2-layer DQNs for the Gridworld 3x3 domain to converge to the optimal policy when the ground state is one-hot encoded, more than 20 are necessary when it is encoded via features.



(a) Gridworld 3x3 (F). (b) Gridworld 3x3 (OH).

Figure 3.24: Mean converged c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV}, c_E and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the Gridworld 3x3 (F) and Gridworld 3x3 (OH) domains. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

When the ground state is encoded via features for our domains, a DQN cannot group any states with the same Q-values from different ground states together in the first layer without mapping states with different Q-values to the same cluster. Hence, a state representation that groups states together if and only if they correspond to the same ground state is optimal for the first network layer for this form of ground state encoding for Gridworld 3x3. With optimal here we mean that the largest amount of state space compression is obtained that is possible without losing any information necessary to learn the Q-values. Moreover, due to the grid structure present in this form of state encoding and the fact that merely linearly separable functions can be learned, the only learnable state representations that groups states together if and only if they belong to the same ground state are ones that closely mirror the underlying grid structure. The first-layer representation is thus hardly changed during training. This is mirrored by the fact that c_E has a value *close to* 1 in the first network layer at the end of training when encoding the ground state via features (see Figure 3.24a). Notice that c_E is not *equal to* 1 at the end of training, because the DQN does learn to map states that differ solely in the superfluous feature value to activations that are more similar than at the beginning of training[35].

4-layer DQN. As the first layer cannot form a state representation that is similar to the coarsest Markov state representation if the ground state is encoded via features, one might expect later hidden layers to create such an internal state representation during training if a DQN has multiple hidden layers. Yet, as depicted in Figure 3.25 for a 4-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (F)(N) domain, this does not occur. While state representations in later hidden layers do become more similar to the coarsest Markov state representation during training than the first-layer representation, none of them becomes more similar to the coarsest Markov state representation than the one in the output layer at any point during training. Instead, the representations in all hidden layers remain very similar to the one in the input layer. This is the case even when one reduces the hidden layer size and thus the network capacity (see Figure 3.26)[36]. The reason for this phenomenon likely is the fact that the initial state representation in each of the hidden layers is very similar to the one in the input layer (see Figure 3.25). Thus,

[33] When grouping states based on bisimilarity or Q-values, all states corresponding to ground states with the same sequence of actions to a goal state under π^*, whose activations are drawn in the same color in t-SNE plots, fall into the same cluster for the Gridworld 3x3 domain.

[34] We provide supporting evidence based on the FrozenLake 4x4 domain in Figure A.15 in the Appendix.

[35] A consequence of this is that c_E typically is higher at the end of training if all features are scaled to $[0, 1]$ than if the features are not scaled. The reason is that states differing merely in the superfluous feature value have encodings with relatively smaller Euclidean distances when all features are scaled to $[0, 1]$ in the (F)(N) state encoding. This is mirrored by the mean converged first-layer value for c_E in Figure A.14 in the Appendix.

[36] Supporting evidence based on the FrozenLake 4x4 domain is given in Figure A.17 in the Appendix.

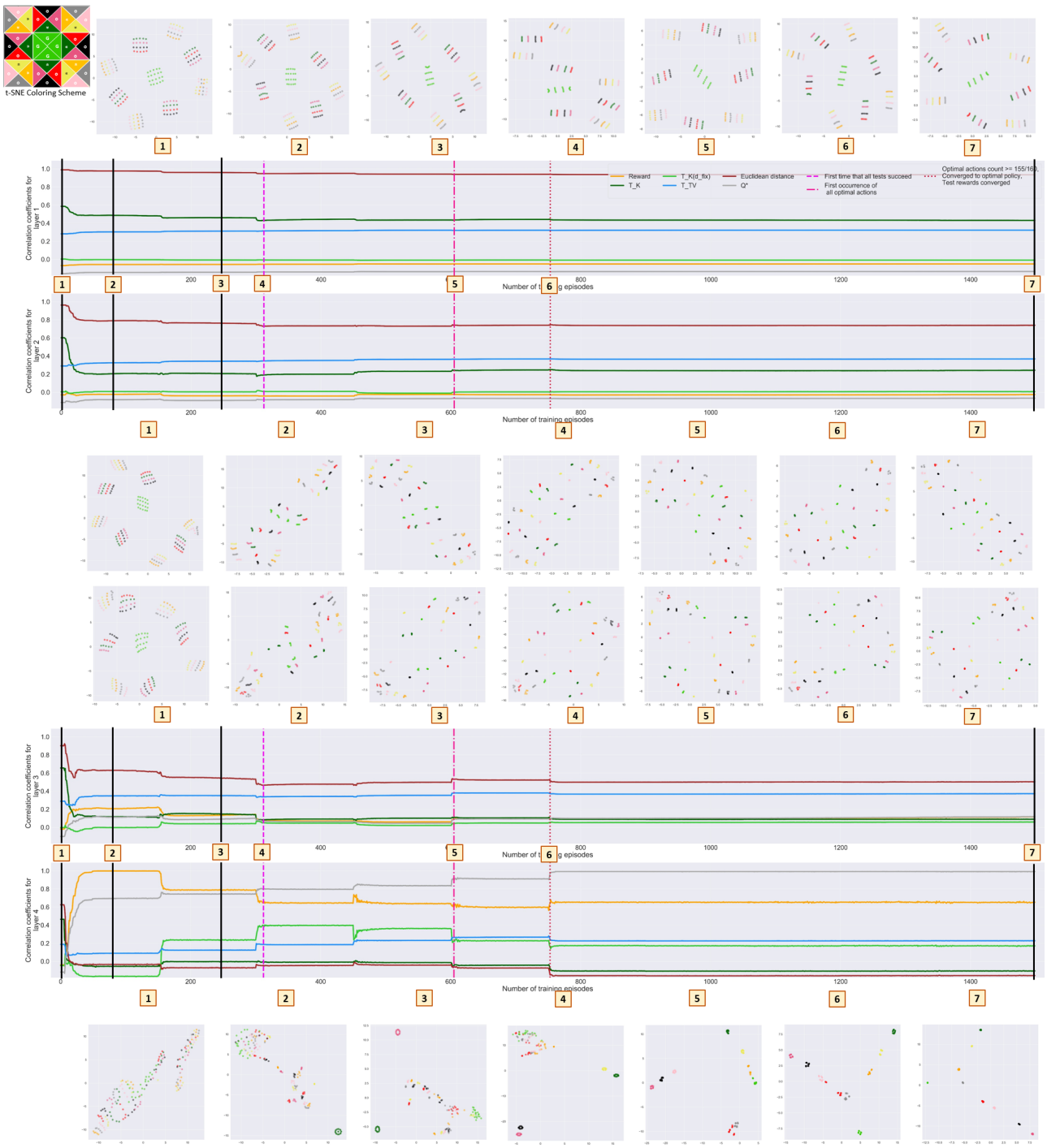


Figure 3.25: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 4-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (F)(N) domain. The coloring scheme is shown in the top-left corner.

just like the first layer cannot group any bisimilar states or states with the same Q-values from different ground states together without grouping some non-bisimilar states or states with different Q-values together, this also holds for the other hidden layers, albeit to slightly lower degrees for later hidden layers. Clustering states primarily based on their ground states therefore poses a local optimum for all three hidden layers of 4-layer DQNs for the Gridworld 3x3 domain when the ground state is encoded via features.

- **Hidden layer size**. The hidden layer size may not only be so large that learning close to the coarsest Markov state representation in the first layer is not necessary, but it may also be too small

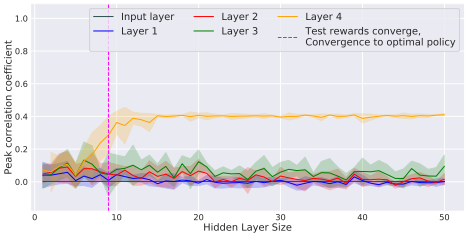


Figure 3.26: Mean peak $c_{K(d_{fix})}$ in each layer with 95%-confidence intervals for each hidden layer size and constant input-layer $c_{K(d_{fix})}$ for a 4-layer DQN for the Gridworld 3x3 (F) domain. Values are based on 5 repetitions. The vertical line indicates the smallest hidden layer size for which the test rewards converge and the network converges to the optimal policy at least one out of 5 times.

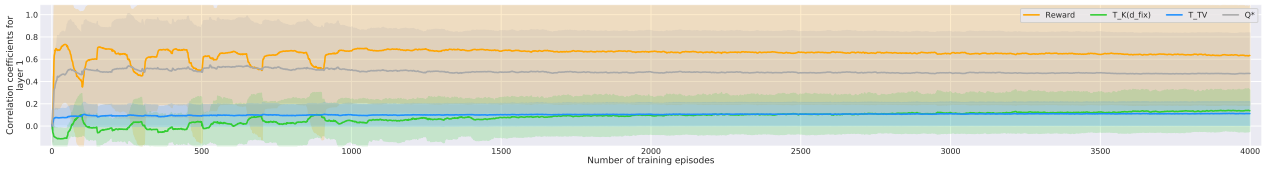
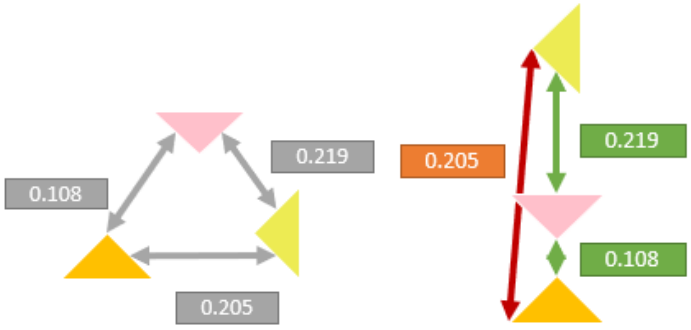


Figure 3.27: Mean c_{Rew}, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for the first layer of a 2-layer DQN with a hidden layer size of 1 for the Gridworld 3x3 (OH) domain. Values are based on 5 repetitions and 95%-confidence intervals are shown.

for it to be possible to create such a first-layer representation. For instance, our experiments for the Gridworld 3x3 (OH) domain show that for a hidden layer size of 1, the first-layer representation does not get significantly more similar to the coarsest Markov state representation during training (see Figure 3.27). As visualized in Figure 3.28, forming the coarsest Markov state representation in the first layer is not possible for a hidden layer size of 1. While it would be feasible to map states to the same first-layer activation pattern if and only if they are bisimilar, it is not viable to create the correct inter-cluster distances for this domain. Similar arguments hold for learning a Q^*-irrelevance abstraction in the first layer. The result is that states in the first layer are largely grouped based on immediate rewards at the end of training. This is mirrored in Figure 3.27 by the high value for the reward-based correlation coefficient c_{Rew} at the end of training.



(a) (b)

Figure 3.28: a) Pairwise distances based on d_{fix} for the first three ground states of the Gridworld 3x3 domain. b) For a hidden layer size of 1, mapping these three states to activations such that the pairwise Euclidean distances of the activations match the pairwise distances based on d_{fix} is not feasible.

Further factors. Besides the factors discussed above, we also investigate the effects of other aspects in the Appendix. We show the impact of optimization settings such as the batch size and the learning rate in Sections C.4.2 and C.4.3. Moreover, factors that do not have a significant impact on the learned hidden-layer state representations are the frequency with which the target network is updated during training as discussed in Section C.4.1, whether or not a fixed replay memory is utilized during training as delineated in detail in Section C.1, and whether a DQN converges to the true Q-values or solely to the optimal policy as analyzed in Section C.5.

4

Impact of Markovianity on Learning Speed and Consistency

In Chapter 3, we saw that internal state representations that are similar to the coarsest Markov state representation tend to be learned in the hidden layers of DQNs during the second learning phase. Intuitively, creating such internal state representations should be useful, because the coarsest Markov state representation distinguishes states if and only if doing so is relevant for predicting rewards and next states. Hence, forming the coarsest Markov state representation at the beginning of training helps both to discard irrelevant information and to maintain sufficient information to represent the transition function and rewards, and should therefore allow for faster and more data-efficient learning. Yet, our results from Chapter 3 also suggest that such a representation may not be created if this is less necessary due to the network capacity. The purpose of this chapter thus is to explore whether forming a hidden-layer representation that is more similar to the coarsest Markov state representation during training for such settings can allow deep RL agents to learn more quickly and more consistently.

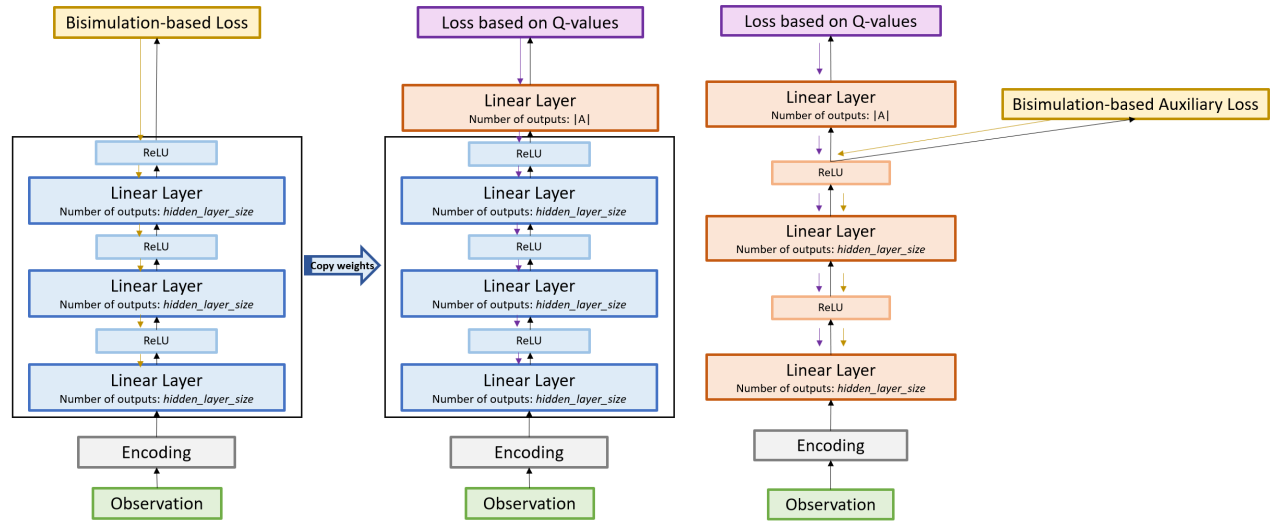
Several options exist for making a deep RL agent learn a hidden-layer state representation that is closer to the coarsest Markov state representation as shown in Figure 4.1:

1. *Pretraining*. Pretraining could strive to create the coarsest Markov state representation in the last layer of a neural network, after which one adds one more layer for learning the Q-values. The pretrained layers could also be reused for domains that have different Q-values, as long as a subset of the relevant[1] features of the original domain are relevant for the related domain. This latter idea forms the basis of some of our experiments in Chapter 5, which examine whether creating a hidden-layer representation that is more similar to the coarsest Markov state representation indeed facilitates transfer to such related domains.

2. *Auxiliary loss.* A bisimulation-based auxiliary loss, which forces a network to learn close to the coarsest Markov state representation in its last hidden layer, could be incorporated into the training of a DQN. This approach is similar to the one of pretraining, except that the steps of forming close to the coarsest Markov state representation in the second-to-last network layer and learning the Q-values are combined. One possible advantage of this combination is that if one is not interested in transfer to related domains, learning the coarsest Markov state representation in the second-to-last network layer solely to some extent may already improve upon the training, while being less time consuming than doing so completely.

In the following, we describe an auxiliary loss that forces a DQN to learn a hidden-layer representation that is similar to the coarsest Markov state representation in Section 4.1, and analyze the extent to which such an auxiliary loss can improve upon the learning of deep RL agents on single domains in Section 4.2. Note that the purpose of this chapter is not primarily to design a novel auxiliary loss, but to see whether it is useful to create internal state representations that are more similar to the coarsest Markov state representation during training. In fact, our best-performing auxiliary loss is in its current

[1] Recall that we define relevant features as those that are needed to predict the reward and relevant features of next states.



(a) (b)

Figure 4.1: Options for using bisimulation metrics to force DQNs to learn internal state representations that are more similar to the coarsest Markov state representation. a) *Pretraining*. A network is pretrained to form close to the coarsest Markov state representation in its last layer. b) *Auxiliary loss*. A bisimulation-based auxiliary loss is computed during training, which impacts the state representations learned in all but the last layer of a DQN.

form not scalable to large domains, as it requires computing the exact bisimulation metric d_{fix}[2] for all pairs of states. While we do explore the impact of employing an approximation of this bisimulation metric instead, doing so comes at the expense of reduced effectiveness. Lastly, even though we do not investigate the impact of pretraining based on bisimulation metrics, an auxiliary loss can also be employed for pretraining by using it as the sole loss. Hence, the auxiliary loss we propose is applicable to the context of pretraining as well.

4.1. Methodology

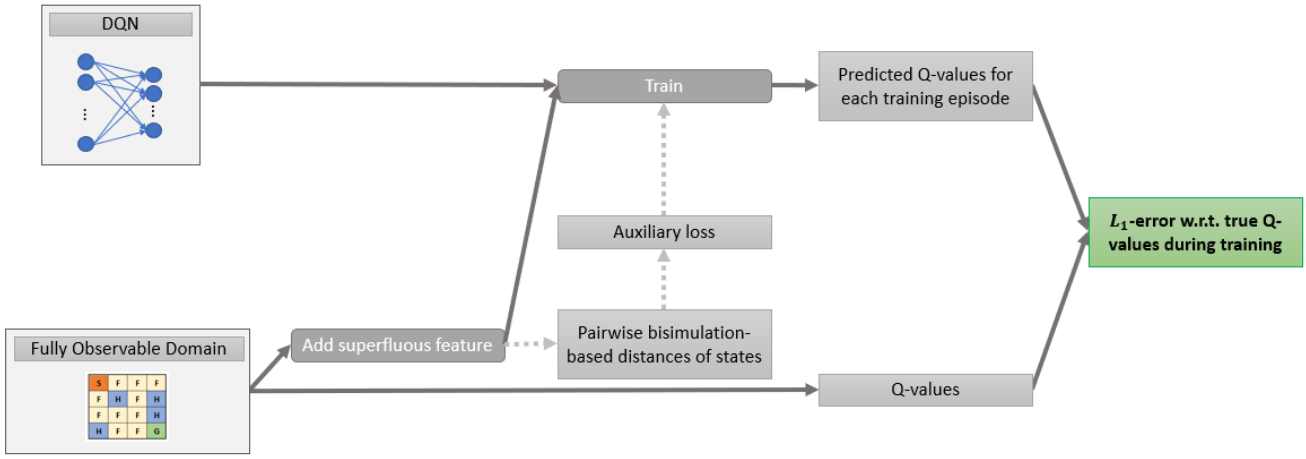


Figure 4.2: Flow-chart of the experiments we conduct to examine the impact of Markovianity on learning speed and consistency.

We empirically elucidate whether forming a hidden-layer representation that is more similar to the coarsest Markov state representation *during* training can allow deep RL agents to learn more quickly and more consistently on a single domain. To this end, we propose an auxiliary loss, which pushes a network to create an internal state representation that is similar to the coarsest Markov state representation in its last hidden layer. As baseline serve 2-layer DQNs trained like the ones in Chapter 3. A flow chart visualizing the experiments conducted for this research question is given in Figure 4.2. Next, we

[2]Refer to Section 2.7.1 for more information on this bisimulation metric.

discuss the auxiliary loss in Section 4.1.1 and the experimental setup in Section 4.1.2.

4.1.1. Auxiliary Loss

Adding auxiliary tasks to the learning process introduces a bias which pushes the agent to learn a state representation that is useful for several tasks, hence reducing the variance and potentially the degree of overfitting to the training data [18]. Our auxiliary loss more specifically forces the Euclidean distance between the activations two states are mapped to in the last hidden layer to be proportional to a distance that is based on bisimulation metrics.

Bisimulation-based measures. We compare the impacts of utilizing multiple distances that are based on or equal to bisimulation metrics. Specifically, we perform experiments based on d_{fix}, T_{TV} and d'_{fix}[3]. While only employing d_{fix} pushes a network to learn the coarsest Markov state representation in its last hidden layer, T_{TV} and d'_{fix} are much faster to compute in practice. In addition, T_{TV} is also independent of the discount factor and the reward function. Lastly, future work should further explore the impact of utilizing d_{TV} as measure for an auxiliary loss, because, unlike T_{TV} and similarly to d_{fix}, it incorporates the reward function but is at the same time much faster to compute exactly than d_{fix}.

Calculation of the auxiliary loss. To calculate the auxiliary loss during training, the pairwise bisimulation metric-based distances d_B between (a subset of) states are computed before training. For each update of the network during training, the auxiliary loss is then calculated by comparing the current Euclidean distance $d_E^{curr}(z_i, z_j)$ between the activations z_i and z_j that states s_i and s_j are mapped to in a hidden network layer to the corresponding target Euclidean distance $d_E^{target}(z_i, z_j)$[4]. For each network update, the steps performed to incorporate the auxiliary loss are as follows:

1. $d_E^{target}(z_i, z_j)$ is computed based on the premise that we want a Pearson correlation coefficient of 1 between d_B and d_E, which means that the points $\big(d_E(z_i, z_j), d_B(s_i, s_j)\big)$ should fall onto a straight line as shown in Figure 4.3. One known point on this line is the point $(0, 0)$, since we would like that $d_E(z_i, z_j) = 0$ if and only if $d_B(s_i, s_j) = 0$. We obtain a second point on this line by determining d_E^{target} for the largest possible value d_B^{max} for d_B[5]. Based on this constructed line, $d_E^{target}(z_i, z_j)$ is computed as so:

$$d_E^{target}(z_i, z_j) = \frac{d_E^{max}}{d_B^{max}} \times d_B(s_i, s_j). \tag{4.1}$$

   Note that d_E^{max} is a hyperparameter that can be tuned.

2. Based on the current and target Euclidean distances of the current activations z, the target activation z_i^{target} is calculated for each state s_i as follows[6]:

$$z_i^{target} = z_i + \frac{1}{2} \times \sum_{j \neq i} \big(d_E^{target}(z_i, z_j) - d_E^{curr}(z_i, z_j)\big) \frac{z_i - z_j}{||z_i - z_j||}, \tag{4.2}$$

   where $||z_i - z_j||$ is the length of the vector $z_i - z_j$[7]. Note that the unit-length vector $\frac{z_i - z_j}{||z_i - z_j||}$ between the current activations of states s_i and s_j is multiplied by half of the amount by which the Euclidean

[3] See Section 3.1.1 for details on how we calculate these measures.

[4] We also performed some experiments based on directly optimizing the Pearson correlation coefficient rather than explicitly calculating target Euclidean distances. This has the advantage that the resulting auxiliary loss is less restrictive. Yet, an auxiliary loss based on this computation did not significantly improve upon the learning of 2-layer DQNs for the FrozenLake 4x4 (OH) domain.

[5] Notice that d_B^{max} is equal to 1 for T_{TV} for all of our domains, whereas it is not clear what d_B^{max} is for d_{fix} or d'_{fix}. One option is to set d_B^{max} equal to the highest occurring value for d_{fix} or d'_{fix} for a specific domain, another one is to utilize a fixed value for all domains.

[6] Instead of calculating the target activations and subsequently minimizing the *Mean Squared Error* (MSE) between current and target activations, one could also directly minimize the MSE between the current and target Euclidean distances of the activations.

[7] To prevent exploding gradients during training, it may be useful to divide the sum in the above equation by the number of other states j or alternatively use a rather low weight for the auxiliary loss.

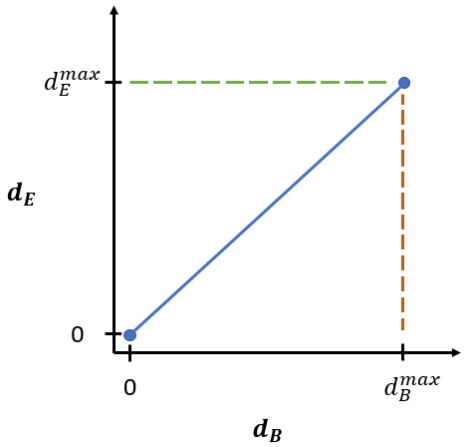


Figure 4.3: Setup for the computation of the target Euclidean distance $d_E^{target}(z_i, z_j)$ for the activations z_i and z_j that the states s_i and s_j are mapped to in a network layer based on $d_B(s_i, s_j)$. The blue line is the line that all points $(d_E(z_i, z_j), d_B(s_i, s_j))$ should fall on to achieve a Pearson correlation coefficient of 1.

distance between the activations of the two states should be changed. The idea behind this is that if the activations of two states should be pulled apart or moved closer together, each activation is moved by half the required total amount in the corresponding direction.

3. The goal is to minimize the MSE between the current and target activations for states $s_i \in S$:

$$MSE(z_i, z_i^{target}) = \sum_{d=0}^{D-1} (z_{i,d} - z_{i,d}^{target})^2, \tag{4.3}$$

where D is the size of the network layer.

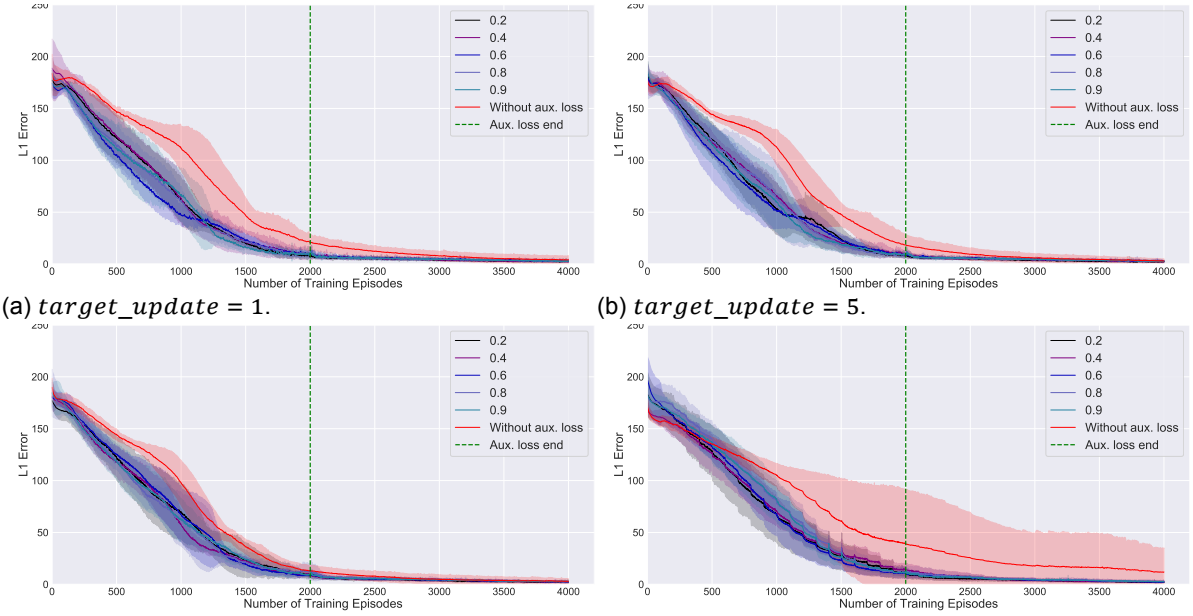
4.1.2. Experiments

To evaluate the effectiveness of this auxiliary loss, we train 2-layer DQNs with varying hidden layer sizes, weights and durations[8] for the auxiliary loss, values for d_E^{max} and target network update frequencies for the FrozenLake 4x4 (OH) and Gridworld 3x3 (OH) domains and measure the L_1-error with respect to the true Q-values for each training episode. We use the (OH) state encoding, as the hidden-layer state representations tend to become less similar to the coarsest Markov state representation during training for this type of state encoding than for other forms of state encoding as described in Section 3.2.2.3. Varying hidden layer sizes are employed, because the network capacity impacts the extent to which the formed first-layer representation is similar to the coarsest Markov state representation when no auxiliary loss is used, as shown by our previously discussed experimental results in Chapter 3. Furthermore, we conduct experiments with different target network update frequencies, since especially when the target network is updated very frequently, adding a bisimulation-based auxiliary loss may be useful to increase learning stability. Lastly, despite the fact that the tested domains are relatively small with state space sizes of 80 and 180 with the added superfluous feature, results for these domains should nevertheless indicate whether or not a bisimulation-based auxiliary loss can improve upon the training of DQNs. Yet, improvement is expected to be more pronounced for larger domains. Details on how we set d_B^{max} for d_{fix} and d'_{fix} can be found in Section A.2.1 in the Appendix.

4.2. Results

We obtain the best results by utilizing an auxiliary loss based on d_{fix}, which forces a DQN to form the coarsest Markov state representation in its last hidden layer. This auxiliary loss can help DQNs for the FrozenLake 4x4 (OH) domain to converge to the true Q-values more quickly as well as to learn more accurate Q-values if the hyperparameters for training without the auxiliary loss otherwise do not allow the DQN to converge to the true Q-values. For example, Figure 4.4 shows that a 2-layer DQN with a

[8] Since the auxiliary loss is expensive to compute, we apply the auxiliary loss with a fixed decay rate of 0.9999 and vary the number of training episodes during which we use the auxiliary loss.



(a) $target\_update = 1$. (b) $target\_update = 5$.

(c) $target\_update = 10$. (d) $target\_update = 100$.

Figure 4.4: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss and varying target network update frequencies. The usage of the auxiliary loss is ended after training episode $2,000$, which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{256 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

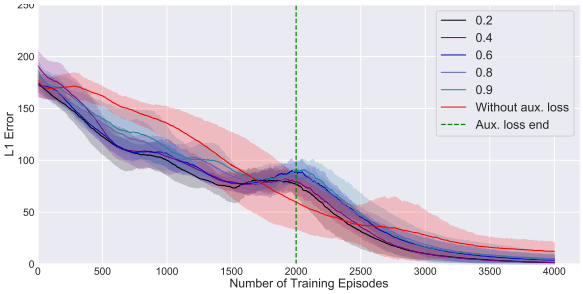
hidden layer size of 50 for the FrozenLake 4x4 (OH) domain converges to the true Q-values almost twice as fast and more reliably when the auxiliary loss is added to the training process for different target network update frequencies. This suggests that creating a hidden-layer state representation that is more similar to the coarsest Markov state representation during training is indeed useful. Supporting evidence based on the Gridworld 3x3 domain is given in Section A.2.2.2 in the Appendix. Furthermore, the impacts of different target network update frequencies and hyperparameters for the auxiliary loss are analyzed in Section A.2.2.1 in the Appendix using the FrozenLake 4x4 domain as example.

Hidden layer size. Introducing the auxiliary loss leads to less improvement in the L_1-error at the beginning of training for smaller hidden layer sizes than for larger ones, even when choosing the best value for d_E^{max} for each hidden layer size (see Figure 4.5). This intuitively makes sense, as a DQN with a lower capacity already forms a hidden-layer representation that is more similar to the coarsest Markov state representation during training when no auxiliary loss is used, and needs to learn closer to a Q^*-irrelevance abstraction in the first layer for the second layer to be able to learn the true Q-values.

Auxiliary loss based on d'_{fix} or T_{TV}. Replacing d_{fix} by the much faster to compute d'_{fix} or T_{TV} also improves the L_1-error at the beginning of training. Yet, whereas using an auxiliary loss based on d_{fix} also allows for significantly earlier convergence to the true Q-values, this is not the case when d'_{fix} or T_{TV} are employed. In contrast to d_{fix}, d'_{fix} does not assign distances of exactly 0 to bisimilar states, which makes it more difficult to learn not to distinguish states based on the superfluous feature value as well as to group bisimilar states from different ground states together while the auxiliary loss is applied. The latter aspect also holds for T_{TV}, as clustering states based on T_{TV} leads to distinct clusters for bisimilar states from different ground states, such as the terminal states in the FrozenLake 4x4 domain. Therefore, our auxiliary loss can be made more scalable by utilizing bisimulation-based measures that do not require calculating the precise Kantorovich distance-based bisimulation metric. Yet, doing so comes at the cost of reduced effectiveness, which underlines the usefulness of creating the precise coarsest Markov state representation rather than approximate or finer versions during training[9]. De-

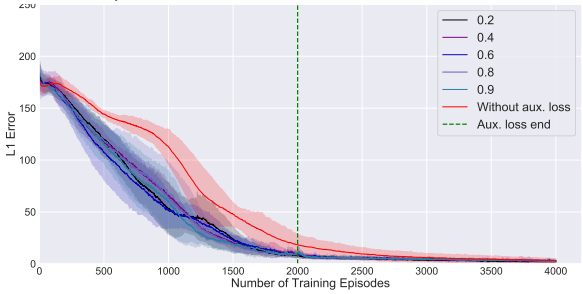
[9]Recall that clustering states based on T_{TV} leads to a strictly finer abstraction of the state space than doing so based on d_{fix} for our domains. The reason is that T_{TV} assigns a non-zero distance to bisimilar states from different ground states for our

$hidden\_layer\_size = 50$.

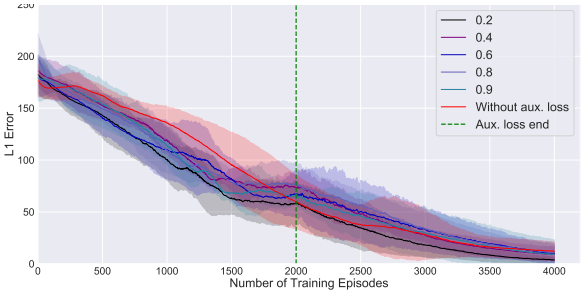


$hidden\_layer\_size = 20$.

(a) $d_E^{max} = \sqrt{32 \times hidden\_layer\_size}$.



$hidden\_layer\_size = 50$.



$hidden\_layer\_size = 20$.

(b) $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$.

Figure 4.5: L_1-error with respect to the true Q-values for a 2-layer DQN for the FrozenLake 4x4 (OH) domain when employing different hidden layer sizes, settings for d_E^{max}, and weights for the auxiliary loss. $target\_update = 5$ and the usage of the auxiliary loss is ended after training episode $2,000$, the latter of which is indicated by the green vertical line. d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

tailed results for the auxiliary losses based on d'_{fix} and T_{TV} are given in Section A.2.3 and Section A.2.4 in the Appendix, respectively.

domains.

5

Impact of Markovianity on Generalization

Chapter 3 showed that the degree to which the formed hidden-layer representations are similar to the coarsest Markov state representation is contingent on several factors such as the network capacity, and that the precise coarsest Markov state representation is not created for any setting in our experiments. Yet, learning an internal state representation that is similar to the coarsest Markov state representation in a DQN's hidden layer by the end of training poses multiple possible advantages over learning a finer or a coarser state representation with regards to generalization:

1. *Transfer to related domains*. A trained network that has formed an internal state representation that is similar to the coarsest Markov state representation could generalize better to related domains with different reward or transition functions, as long as the related domain's relevant features are a subset of the relevant[1] features of the original domain. Notice that changing the transition and reward functions in a way that maintains the abstract states of the coarsest Markov state abstraction is realistic. For instance, consider our introductory example of a firefighter robot in a domain in which the color of an object is irrelevant and observations that differ solely in the object color are hence bisimilar. Plausible related domains are ones in which the immediate rewards for pouring water upon seeing "smoke above blue house" or "smoke above orange house" or the probabilities of extinguishing the fire in those states differ while an object's color remains irrelevant. Depending on the precise transition and reward functions, the states "smoke above blue house" and "smoke above orange house" may or may not have the same Q-values as states such as "burning red car" and "burning black car." This means that a coarser abstraction of the state space than the coarsest Markov state representation that clusters states solely based on Q-values may fail to generalize to related domains. Similarly, if the formed hidden-layer representation does not regard states differing solely in the color of an object as equivalent and is thus finer than the coarsest Markov state representation, more information than necessary is transferred to a related domain. Therefore, more data and time may be required to learn the new Q-values based on the transferred state representation.

2. *Robustness to superfluous feature values unseen during training.* A network that has learned a hidden-layer representation similar to the coarsest Markov state representation could generalize better to irrelevant feature values not encountered during training. For example, using different values for the superfluous feature value during testing than training could have less severe of an impact on the performance if the representations formed in the hidden layers are closer to the coarsest Markov state representation. This is the case, because such representations tend to ignore superfluous features to a larger extent than less Markov state representations. In the firefighter example, for instance, a hidden-layer representation that does not distinguish objects based on their colors allows the agent to act optimally in the state "smoke above yellow house" even if the agent has solely visited the states "smoke above blue house" and "smoke above orange house" during training. Of course, learning a precise Q^*-irrelevance abstraction in a hidden layer would lead to the same generalization performance to superfluous feature values unseen

[1] Recall that we define relevant features as those that are needed to predict the reward and relevant features of next states.

during training as learning the coarsest Markov state representation. The reason is that both of these representations are entirely indifferent to superfluous features. Yet, as discussed above, forming an exact Q^*-irrelevance abstraction in a hidden layer may render generalization to related domains with different transition or reward functions more difficult.

This chapter thus explores the extent to which learning a hidden-layer representation that is closer to the coarsest Markov state representation renders a network trained on one domain more useful for related domains with modified reward or transition functions in Section 5.1, and makes a DQN more robust to new superfluous feature values in Section 5.2.

5.1. Transfer to Related Domains

We outline our methodology in Section 5.1.1 and our results in Section 5.1.2.

5.1.1. Methodology

As depicted in Figure 5.1, we train 2-layer DQNs either with or without the bisimulation-based auxiliary loss described in Chapter 4. Afterwards, we transfer learned internal state representations to DQNs that are then retrained on related domains. The related domains and the transfer of state representations to related domains are described in the following. Details regarding the auxiliary loss we utilize for some experiments to make the hidden-layer representations formed on an original domain more similar to the coarsest Markov state representation are given in Section A.3.1.1 in the Appendix.

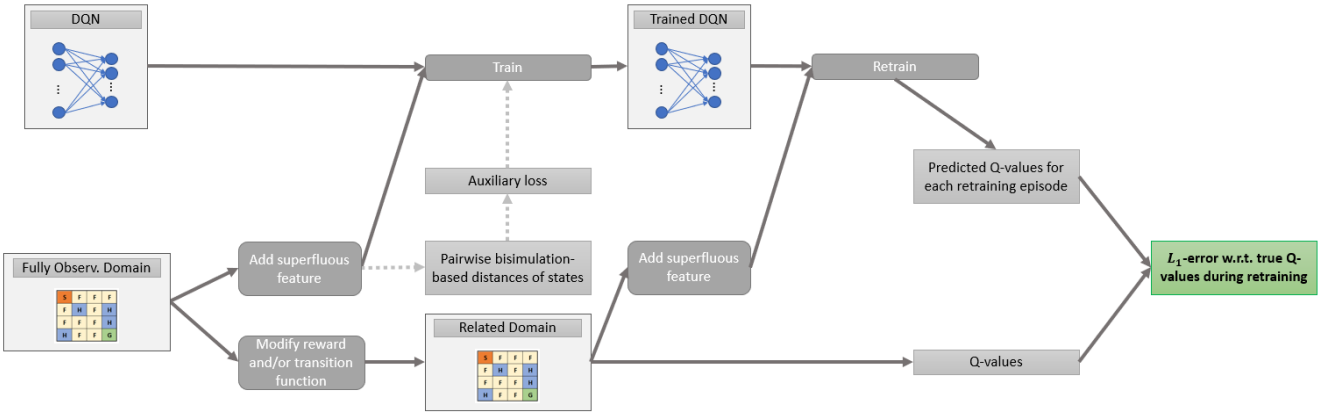


Figure 5.1: Flow-chart of the experiments conducted to explore the impact of Markovianity on generalization to related domains.

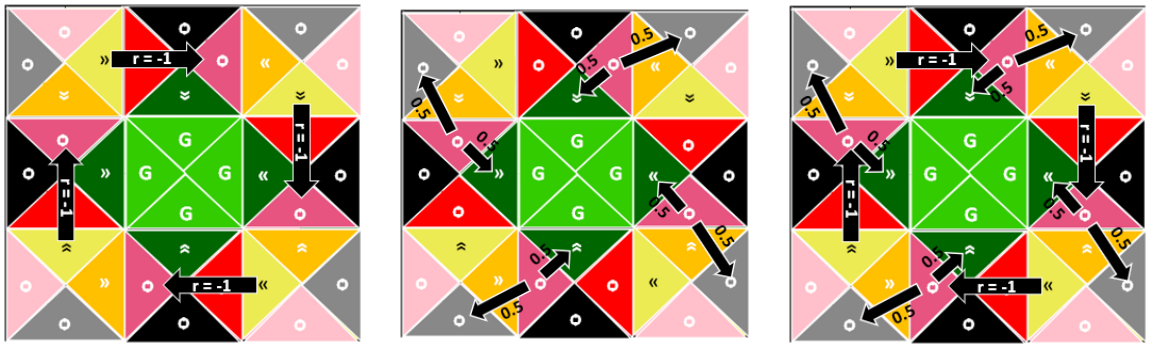
5.1.1.1 Related Domains

We create three altered versions of the Gridworld 3x3 domain, in each of which two states are bisimilar if and only if they are bisimilar in the original domain:

1. *Modified reward function*. We change the reward function by setting the immediate reward for moving forward in four bisimilar ground states to -1 rather than 0. This leads to different optimal actions for those four ground states as well as to different Q-values for the four ground states and all states for which the path under the optimal policy leads through one of the four ground states for some action. The altered domain is depicted in Figure 5.2a and referred to as Gridworld^R 3x3.

2. *Modified transition function*. We alter the transition function for a set of four bisimilar ground states. More precisely, the action $rotate$ for these four ground states no longer causes a deterministic transition, but a stochastic one with two equally likely next ground states. This change leads to different Q-values, but the optimal actions of all states are the same as in the original domain. This related domain is visualized in Figure 5.2b and we denote it by Gridworld^T 3x3.

3. *Modified reward and transition functions*. We combine the two modifications above to yield a domain in which both the Q-values and the optimal actions are different for some states compared to the original domain. This domain is shown in Figure 5.2c and we refer to it by Gridworld^{RT} 3x3.



(a) Gridworld^R 3x3: Modified reward function.

(b) Gridworld^T 3x3: Modified transition function.

(c) Gridworld^{RT} 3x3: Modified reward and transition functions.

Figure 5.2: Domains related to the Gridworld 3x3 domain. The coloring of the ground states is based on the colors we assign to the activations of states in our t-SNE plots to highlight which ground states are bisimilar. A circle in a ground state indicates that the optimal action in the original domain is to rotate clockwise, whereas an arrow denotes that the optimal action in the original domain is to move forward. Black arrows indicate changes in the reward or transition function compared to the original domain for a state's optimal action in the original domain. If the modification is related to the transition function, the black arrow is labeled with the probability of transitioning to a certain next ground state, and if the modification occurs with respect to the reward function, the black arrow is labeled with the new immediate reward r.

In addition, we design two domains related to the FrozenLake 4x4 domain, in each of which two states are bisimilar if and only if they are bisimilar in the original domain. These domains are defined in Section A.3.1.1 in the Appendix. The results of the experiments based on these two domains serve as supporting evidence and can be found in Section A.3.1.3.

5.1.1.2 Transfer to Related Domain

We copy representations learned by the 2-layer DQNs trained on the original domains and retrain them on a related domain. Thereby, we compute the L_1-error with respect to the true Q-values of the related domain for each retraining episode. We distinguish three cases with respect to which weights we transfer to a related domain and which weights are updated during retraining:

1. *All weights are transferred, but only the ones in the last layer are updated during retraining*. In this scenario, the formed first-layer representation allows a DQN to converge to the true Q-values of the related domain only if it distinguishes states based on features that are relevant for learning the Q-values of the related domain. This condition is not met if the first layer has formed a Q^*-irrelevance abstraction for the original domain and it does not hold that states have the same Q-values in the related domain if they have the same Q-values in the original domain.

2. *Only the first-layer weights are transferred and these weights are not updated during retraining*. The motivation for this scenario is that if the Q-values of the related and the original domain are very similar, the differences in L_1-error during retraining may not be very visible for different first-layer representations if the output-layer weights are also copied. Note that for our experiments, states have the same Q-values in the related domain if and only if they have the same Q-values in the original domain. In this scenario, the output-layer weights thus are newly initialized.

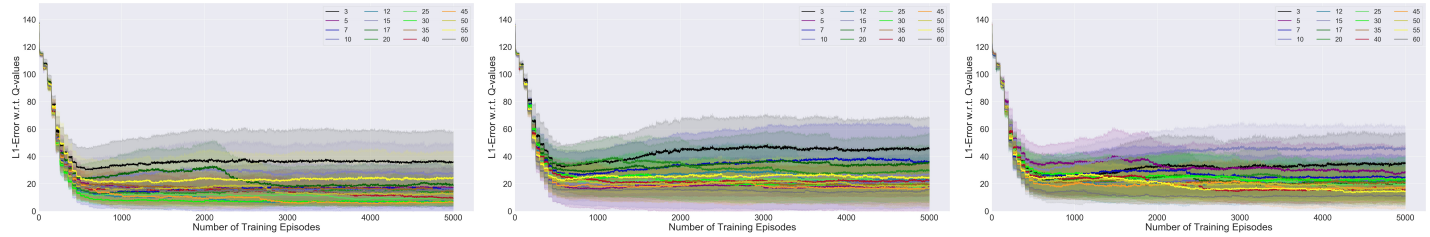
3. *All weights are transferred and updated during retraining*. In this transfer mode, the formed first-layer representation and the learned Q-values are transferred and updated during retraining. The reason for adding this scenario is that for very small hidden layer sizes, creating a first-layer representation that is close to the coarsest Markov representation may not lead to a sufficient compression of the state space for the output layer to converge to the true Q-values during retraining. Hence, the first-layer representation learned for the original domain only provides the initialization for the learning in the related domain in this scenario.

5.1.2. Results

In the sequel, we describe our results for the experiments in which the pretraining does not include the auxiliary loss in Section 5.1.2.1 and those for adding the auxiliary loss to the training on the original domains in Section 5.1.2.2. Thereby, we use the experiments conducted on domains related to

the Gridworld 3x3 domain as primary examples. However, Section A.3.1.3 in the Appendix contains detailed results for the experiments performed for both FrozenLake 4x4 and Gridworld 3x3. In addition, Section A.3.1.2 in the Appendix explores the impact of changing the target Euclidean distances between non-bisimilar states for the auxiliary loss.

5.1.2.1 Pretraining Without Auxiliary Loss



(a) Gridworld 3x3^T (OH). (b) Gridworld 3x3^T (OH)(N). (c) Gridworld 3x3^T (OH)(F-OH).

Figure 5.3: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the Gridworld^T 3x3 domain. The weights from solely the first network layer are transferred and those are not updated during retraining. Values are based on 10 repetitions and 95%-confidence intervals are shown.

Figure 5.3 reveals based on the Gridworld^T 3x3 domain that the following factors impact how well a 2-layer DQN generalizes to a related domain:

1. The generalization performance is better when the formed first-layer representation is *less close to a Q^*-irrelevance abstraction* for the original domain. Recall that DQNs with larger hidden layers learn first-layer representations that are less similar to a Q^*-irrelevance abstraction[2], which explains why DQNs with larger hidden layer sizes tend to generalize best for the (OH)(N) and (OH)(F-OH) state encodings. In addition, the created first-layer representations for larger-than-necessary hidden layer sizes are closer to a Q^*-irrelevance abstraction for the (OH)(N) and (OH)(F-OH) state encodings than for the (OH) encoding, which is why the former lead to higher L_1-errors on the related domain.

2. Lower L_1-errors on the related domain are achieved when the formed first-layer representation is *closer to the coarsest Markov state representation*. Moderately sized hidden layers are more similar to the coarsest Markov state representation for the (OH) encoding than even larger hidden layers, which is why the former lead to better generalization performance than the latter (see Figure 5.3a)[3]. For the (OH)(N) and (OH)(F-OH) forms of state encoding, the largest tested hidden layer sizes do not yet cause the first-layer representations to be less similar to the coarsest Markov state representation by the end of training[4]. Hence, DQNs with moderately sized hidden layers do not outperform DQNs with even larger hidden layers for these two types of state encoding.

3. DQNs with *larger hidden layers* are less dependent on the first-layer representation when it comes to learning the new Q-values. This adds to the fact that DQNs with larger hidden layers generalize best for the (OH)(N) and (OH)(F-OH) forms of state encoding. Notice also that DQNs with very small hidden layer sizes such as 3 need to learn a first-layer representation that is more similar to a Q^*-irrelevance abstraction based on the modified reward and transition functions to be able to converge to the true Q-values. This is not possible if the first layer's weights are fixed during retraining.

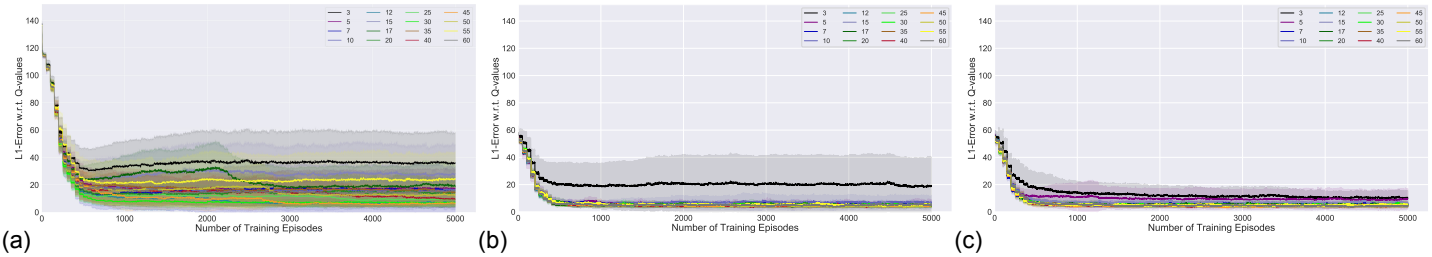
Thus, learning a hidden-layer representation that is similar to the coarsest Markov state representation tends to be beneficial for generalization, especially for moderately sized hidden layers.

When a 2-layer DQN is trained without auxiliary loss on an original domain, we further find that the following factors impact how useful a hidden-layer representation that is similar to the coarsest Markov state representation is for generalization:

[2] Refer to Section 3.2.2.3 for more information on the impact of the hidden layer size and the state encoding on the formed hidden-layer state representation.

[3] Section 3.2.2.3 discusses the impact of the (OH) state encoding on the hidden-layer state representation. Also refer to Figure A.28c in the Appendix to see how similar to the coarsest Markov state representation the final first-layer state representation of a 2-layer DQN is for different hidden layer sizes for the (OH) state encoding.

[4] This is depicted in Figures A.28a and A.28b in the Appendix.



(a) (b) (c)

Figure 5.4: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the Gridworld^T 3x3 (OH) domain. Values are based on 10 repetitions and 95%-confidence intervals are shown. a) The weights from solely the first network layer are transferred and those are *not* updated during retraining. b) The weights from both network layers are transferred, but solely those from the last layer are updated during retraining. c) The weights from both network layers are transferred and updated during retraining.

1. **Whether or not the learned output-layer representation is transferred**. The extent to which the formed first-layer representation is similar to the coarsest Markov state representation has no significant impact on the generalization performance if the last layer's weights are also transferred. Figure 5.4 visualizes that all sufficiently large hidden layer sizes allow DQNs trained without the auxiliary loss on the original domain to generalize similarly well in that case. This intuitively makes sense, as states in our modified domains have the same Q-values if and only if they have the same Q-values in the original domain. Hence, transferring the output layer's weights together with the ones from the input layer implies that the correct abstract states are already present in the output layer at the beginning of training, and that solely some inter-cluster distances need to be adjusted to represent the modified Q-values. This also means that whether or not the first layer's weights are updated during retraining does not make a significant difference when the output layer's weights are transferred, except for very small hidden layer sizes (see Figures 5.4b and 5.4c). Recall that DQNs with smaller hidden layer sizes such as 3 or 5 need to create closer to a Q^*-irrelevance abstraction based on the modified domain's reward and transition functions in the first layer to be able to learn accurate Q-values.
2. **Way in which the related domain differs from the original one**. Overall, the Gridworld^T 3x3 domain is the related domain to which DQNs generalize worst if the last layer's weights are not transferred (see Figure 5.5). It is also for this domain that it becomes clear that DQNs with a more Markov first-layer state representations such as those with a hidden layer size of 30 generalize better than even larger DQNs for the (OH) type of state encoding. Yet, such small differences in the first-layer representations do not have a significant impact on the generalization performances of DQNs on other related domains. This is mirrored in Figures 5.5b and 5.5c, where all DQNs with sufficiently large hidden layers learn similarly well for Gridworld^R and Gridworld^{RT} 3x3.

(a) Gridworld^T 3x3 (OH). (b) Gridworld^R 3x3 (OH). (c) Gridworld^{RT} 3x3 (OH).

Figure 5.5: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the (OH) state encoding. The weights from solely the first network layer are transferred and those are *not* updated during retraining. Values are based on 10 repetitions and 95%-confidence intervals are shown.

The likely reason for the much worse generalization performance to the Gridworld^T 3x3 domain is twofold[5]. First, the way in which the transition function is altered introduces stochasticity to the

[5]Modifying the transition function alone is not the reason, because the L_1-errors achieved on the FrozenLake^R 4x4 and FrozenLake^{RT} 4x4 domains when only the first layer's weights are pretrained are similar, as shown in Figure A.52 in the Appendix. The degree to which the Q-values of a modified domain differ from the ones of the original domain also is not the only cause, as the initial L_1-error is much higher for Gridworld^R 3x3 than for Gridworld^T 3x3, and still the final L_1-error at the end of training is highest for Gridworld^T 3x3 (see Figure 5.5).

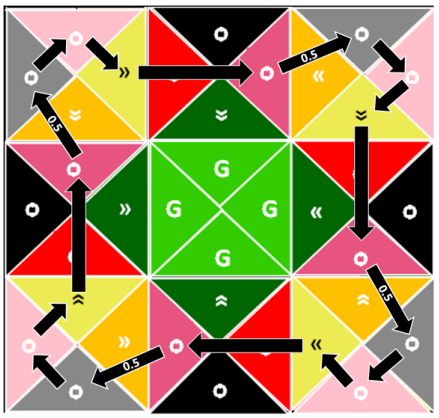


Figure 5.6: Possible path through the Gridworld^T 3x3 domain when following the optimal policy. Stochastic transitions are labeled with their probabilities.

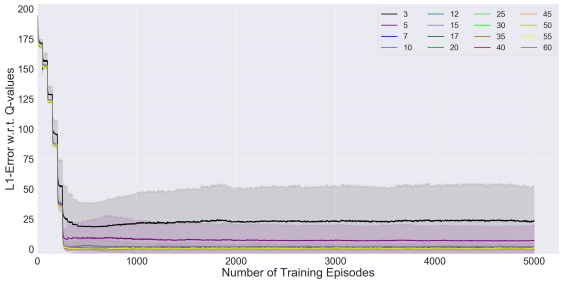
transitions. The result is that with some non-zero probability, the agent goes in multiple circles around the domain without reaching the goal state when following the optimal policy (see Figure 5.6). Consequently, unless the hidden layer can represent the transition function well, it likely is rather difficult to learn the Q-values for this domain. Yet, this specific modification of the transition function cannot be the sole reason why DQNs do not converge to the true Q-values for this domain if solely the first layer's weights are copied. This is the case, because more accurate Q-values are learned on the Gridworld^{RT} 3x3 domain, whose transition function is altered in the same way, albeit full convergence to the true Q-values is not achieved for that domain either. The second contributing aspect therefore appears to be that since the reward function is not modified and the overall difference in Q-values compared to the original domain is comparatively small, a local optimum different from the global one is found for Gridworld^T 3x3. However, as this explanation is not entirely satisfying, it would be interesting to explore in more detail in future work which kinds of modifications to the reward and transition functions render generalization more difficult.

5.1.2.2 Pretraining With Auxiliary Loss

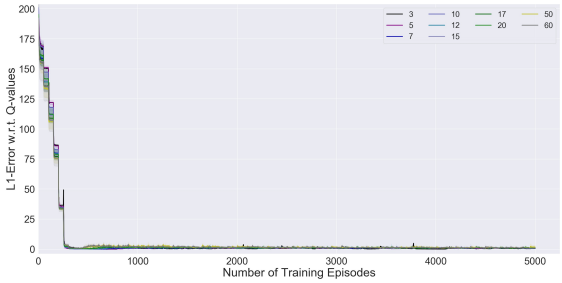
Introducing the auxiliary loss to the training on the original domain generally improves upon the generalization performance of 2-layer DQNs. This suggests that a first-layer representation that is more similar to the coarsest Markov state representation facilitates transfer to related domains[6]. For example, Figure 5.7a shows that now all hidden layer sizes allow DQNs to converge to the true Q-values of the Gridworld^R 3x3 domain. Overall, the improvement is largest for settings for which DQNs trained without auxiliary loss tend to generalize relatively badly. Intuitively this makes sense, because if DQNs with less Markov first-layer representations already generalize well, little improvement is to be expected from making the first-layer representations more Markov. These are the cases for which introducing the auxiliary loss to the training on the original domain has the most noticeable positive effect:

- The largest improvement is obtained for the *Gridworld^T 3x3 domain when the last layer's weights are not transferred*, where all DQNs except those with very small hidden layer sizes now converge to much more accurate Q-values (see Figure 5.7b). Slightly less extensively but still very noticeably lower L_1-errors are achieved on the Gridworld^{RT} 3x3 domain when the last layer's weights are not transferred as depicted in Figure 5.7c.
- Improvements for the transfer scenarios in which the output layer's weights are copied are typically smaller than *if the output layer's weights are not transferred* (see Figure 5.8). Recall that states have the same Q-values in a related domain if and only if they have the same Q-values in the original domain and that the reward and transition functions are modified for solely a few states. Hence, transferring the output layer's weights together with the ones from the input layer means that few changes need to be made to the output-layer representation to represent the new Q-values.

[6] Yet, it matters not only how similar to the coarsest Markov state representation the formed first-layer representation is, but also how large the Euclidean distances between non-bisimilar states are as discussed in Section A.3.1.2 in the Appendix.

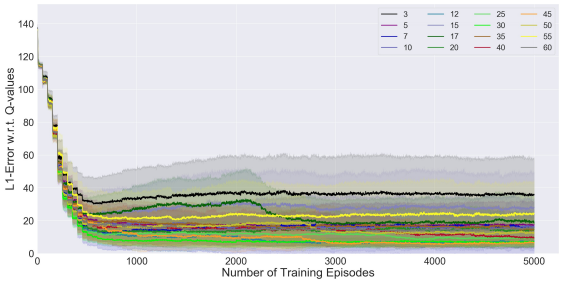


Without bisimulation-based auxiliary loss.

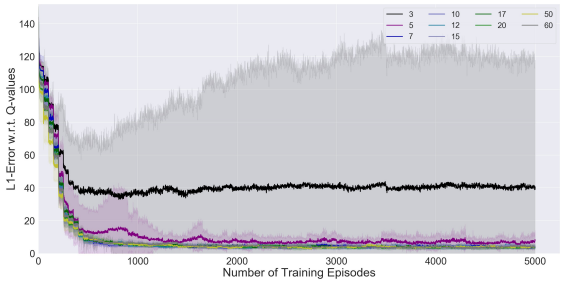


With bisimulation-based auxiliary loss.

(a) Gridworld^R 3x3 (OH).

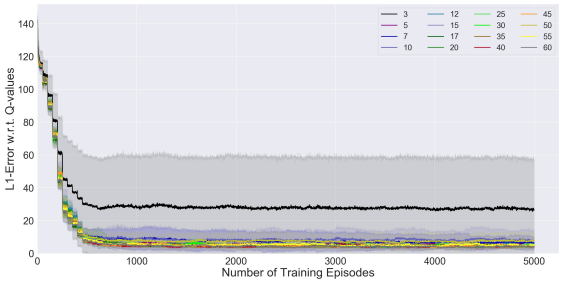


Without bisimulation-based auxiliary loss.

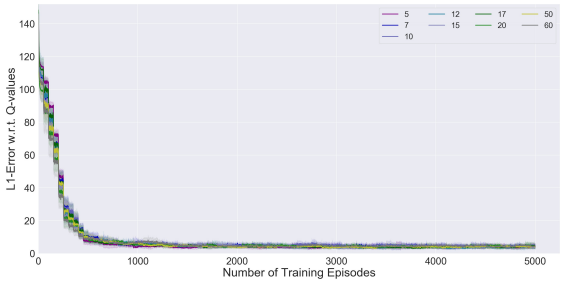


With bisimulation-based auxiliary loss.

(b) Gridworld^T 3x3 (OH).



Without bisimulation-based auxiliary loss.



With bisimulation-based auxiliary loss.

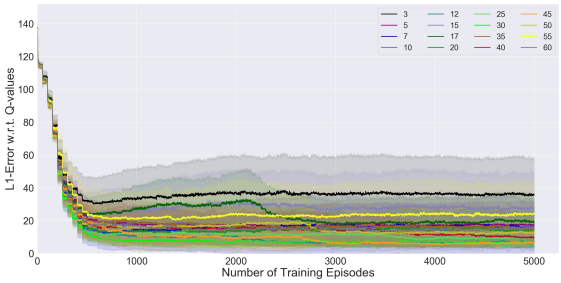
(c) Gridworld^{RT} 3x3 (OH).

Figure 5.7: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with varying hidden layer sizes for the different domains related to the Gridworld 3x3 domain. The first-layer weights are initialized to those of DQNs trained on the Gridworld 3x3 domain either with or without auxiliary loss and are not updated during retraining. The second-layer weights are newly initialized. Values are based on 10 repetitions and 95%-confidence intervals are shown.

- Improvements tend to be larger for *DQNs with relatively small hidden layer sizes* such as 5[7]. Moreover, even when the output layer's weights are transferred, DQNs with smaller hidden layers now sometimes converge more quickly to the true Q-values of the related domain than DQNs with larger hidden layers. For instance, Figures 5.8b and 5.8c depict that if both layers' weights are transferred to the Gridworld^T 3x3 domain, DQNs with a hidden layer size of 5 often converge more quickly than DQNs with larger hidden layer sizes[8]. This confirms that learning an internal state representation that is similar to the coarsest Markov state representation and finer than a Q^*-irrelevance abstraction is useful for generalization purposes. Note that DQNs with small hidden layers learn a first-layer representation that is close to a Q^*-irrelevance abstraction when no auxiliary loss is utilized. Furthermore, when the auxiliary loss is added to the pretraining,

[7] However, since adding the auxiliary loss makes it harder for DQNs with small hidden layers to converge to the true Q-values on the original domain, the L_1-error at the beginning of retraining when the output layer's weights are transferred is sometimes higher than when no auxiliary loss is added. This renders learning on the modified domain more difficult than when the initialization is more favorable.

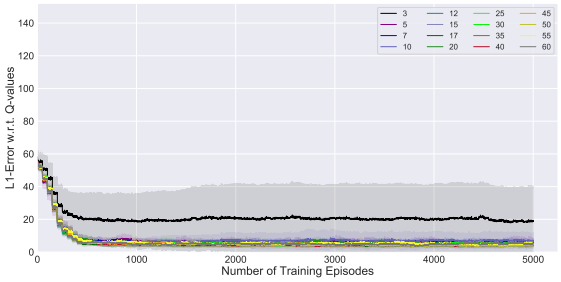
[8] Note that DQNs with a hidden layer size of 3 do not generally converge even sooner than DQNs with a hidden layer size of 5 on the Gridworld^T 3x3 domain. The reason for this is twofold. First, DQNs with a hidden layer size of 3 have formed more similar to a Q^*-irrelevance abstraction in the first layer for the original domain due to our settings for the auxiliary loss as discussed in Section A.3.1.1 in the Appendix. Second, such DQNs need to learn a first-layer representation that is closer to a Q^*-irrelevance abstraction based on the modified transition function, which is not possible if the first layer's weights are not updated during retraining and may not succeed even if the weights are not fixed.



Without bisimulation-based auxiliary loss.

With bisimulation-based auxiliary loss.

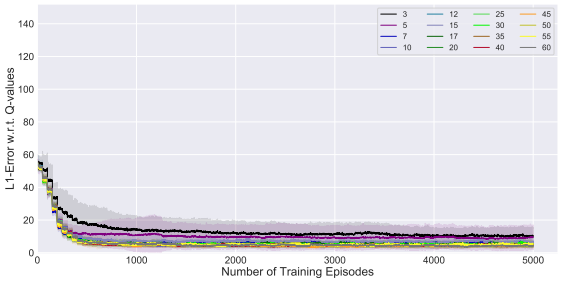
(a)



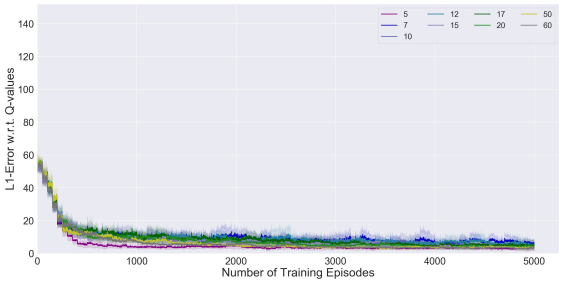
Without bisimulation-based auxiliary loss.

With bisimulation-based auxiliary loss.

(b)



Without bisimulation-based auxiliary loss.



With bisimulation-based auxiliary loss.

(c)

Figure 5.8: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the Gridworld^T 3x3 (OH) domain. Values are based on 10 repetitions and 95%-confidence intervals are shown. a) The weights from solely the first network layer are transferred and those are *not* updated during retraining. b) The weights from both network layers are transferred, but solely those from the last layer are updated during retraining. c) The weights from both network layers are transferred and updated during retraining.

DQNs with slightly larger than necessary hidden layer sizes tend to be initialized with a first-layer representation that is closer to the coarsest Markov state representation than the ones of DQNs with larger hidden layers due to our settings for the auxiliary loss as discussed in Section A.3.1.1 in the Appendix.

5.2. Robustness to Superfluous Feature Values Unseen During Training

We outline our methodology in Section 5.2.1 and our results in Section 5.2.2.

5.2.1. Methodology

To investigate whether forming a hidden-layer representation that is closer to the coarsest Markov state representation aids a DQN in generalizing to superfluous feature values not encountered during training, we train the same 2-layer DQNs as in Chapter 3 with different hidden layer sizes and two forms of state encoding for the Gridworld 3x3 domain. Moreover, to obtain hidden-layer representations that are more similar to the coarsest Markov state representation, we apply the auxiliary loss described in Chapter 4 to the training of some DQNs. For each of these DQNs, we subsequently compute the optimal action returned for each non-terminal ground state when sampling $1,000$ values for the superfluous

feature uniformly at random from an interval that is i times as large and centered at the same value as the interval used during training, where $i \in \{1, 2, 4, 6, 8, 10, 25, 50, 100, 500, 1000\}$. Note that even if $i = 1$, the superfluous feature values generated at test time are not necessarily equivalent to those seen during training, as our networks are trained with only 5 specific values for the superfluous feature. Moreover, notice that the number of generated superfluous feature values decreases relative to the interval size for larger intervals, thus rendering averages less reliable for such intervals. Figure 5.9 shows a flow-chart of our experiments and methodological details are provided in Section A.3.2.1 in the Appendix.

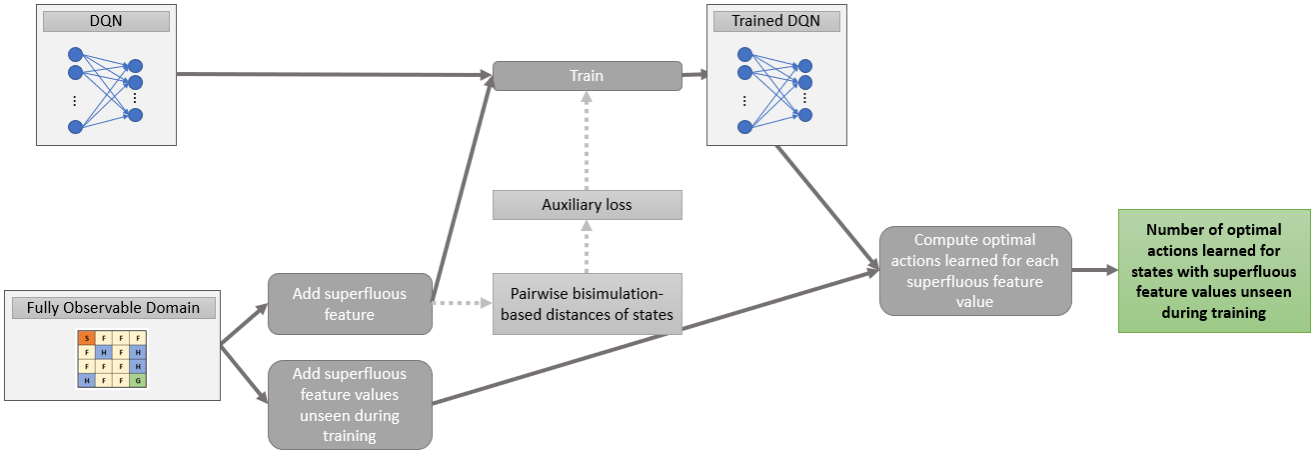
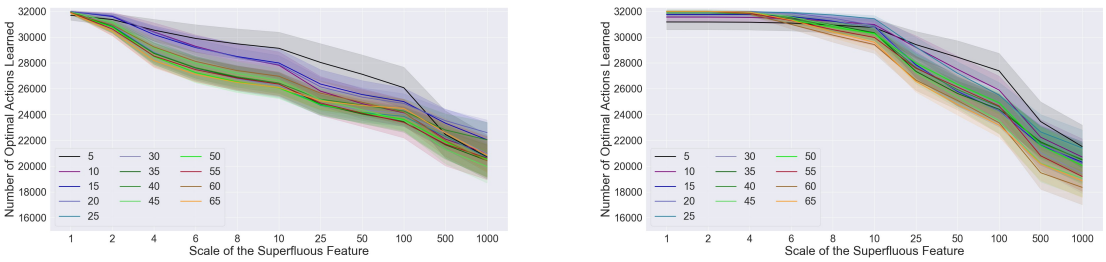


Figure 5.9: Flow-chart of the experiments performed to investigate the impact of Markovianity on generalization to superfluous feature values not encountered during training.

5.2.2. Results

We describe our results based on training DQNs without and with the bisimulation-based auxiliary loss in Section 5.2.2.1 and Section 5.2.2.2, respectively. Furthermore, more detailed results for adding the auxiliary loss to the training process are given in Section A.3.2.2 in the Appendix.

5.2.2.1 Training Without Auxiliary Loss



(a) Gridworld 3x3 (OH). (b) Gridworld 3x3 (OH)(N).

Figure 5.10: Average numbers of optimal actions learned by 2-layer DQNs with different hidden layer sizes. Optimal action returned for each non-terminal ground state are measured when sampling $1,000$ different values for the superfluous feature uniformly at random from an interval that is i times as large as the one used during training. The value i is depicted on the x-axis. Since there are 32 non-terminal states in the Gridworld 3x3 domain, returning $32,000$ optimal actions is optimal. Values are based on 30 repetitions and 95%-confidence intervals are shown.

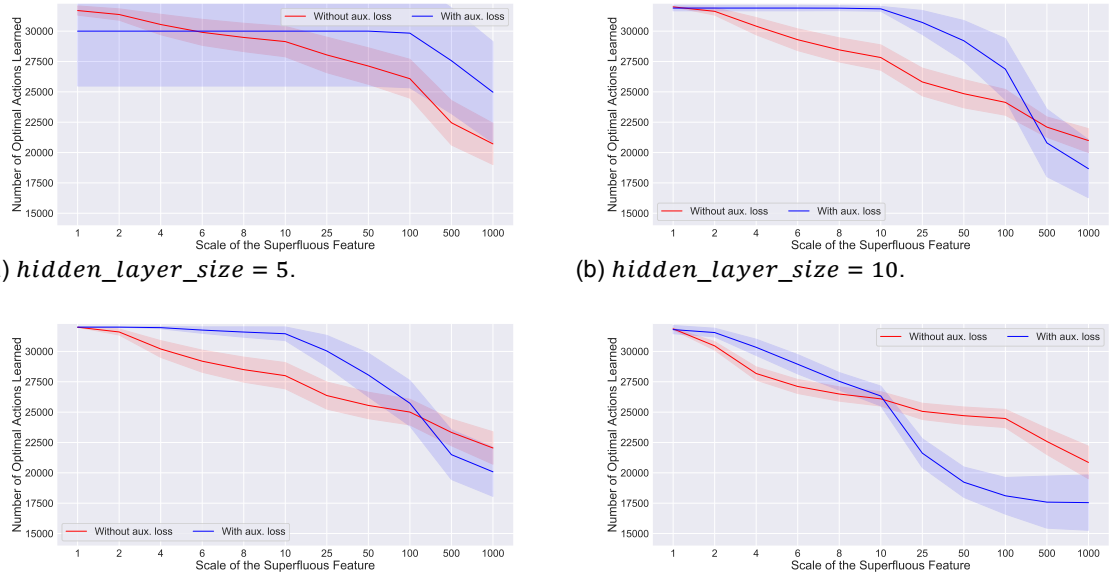
As depicted in Figure 5.10, smaller hidden layer sizes generally allow for better generalization to superfluous feature values unseen during training when no auxiliary loss is added to the training process. Yet, as the final first-layer representations of such DQNs are more similar to both a Q^*-irrelevance abstraction and the coarsest Markov state representation[9], these experimental results allow us to conclude only that coarser abstractions allow for better generalization than finer ones. This observation

[9]This is discussed in Section 3.2.2.3 and depicted in Figure A.28 in the Appendix.

intuitively makes sense, because coarser abstractions tend to ignore the superfluous feature to a higher degree. As mentioned in Section A.3.2.1 in the Appendix, this is not surprising, because networks with large capacities are known to tend to overfit to the training data and hence to generalize worse to data unseen during training. Lastly, notice that small hidden layer sizes such as 5 and 10 do not always enable generalization to all superfluous feature values generated at test time for small intervals, and hence lead to worse generalization than larger hidden layer sizes for such intervals. This pattern is due to the fact that the learning of all true Q-values does not always succeed for such small hidden layer sizes.

5.2.2.2 Training With Auxiliary Loss

Figure 5.11[10], reveals that if a bisimulation-based auxiliary loss is added to the training process, the generalization performance tends to be better than if no auxiliary loss is introduced[11]. This improvement instinctively makes sense, because utilizing the auxiliary loss to push a DQN to learn the coarsest Markov state representation in its first layer typically allows the learned first-layer representation to ignore the superfluous feature to a larger extent. For instance, Figure 5.12 visualizes for a 2-layer DQN with a hidden layer size of 3 that the first-layer representation created without auxiliary loss is finer than the one formed with auxiliary loss when it comes to the activations that states differing solely in the superfluous feature are mapped to.



(a) $hidden\_layer\_size = 5$. (b) $hidden\_layer\_size = 10$.

(c) $hidden\_layer\_size = 15$. (d) $hidden\_layer\_size = 65$.

Figure 5.11: Average numbers of optimal actions learned by 2-layer DQNs for Gridworld 3x3 (OH) with different hidden layer sizes, with and without the auxiliary loss added to the training process. Optimal action returned for each non-terminal ground state are measured when sampling $1,000$ different values for the superfluous feature uniformly at random from an interval that is i times as large as the one used during training. The value i is depicted on the x-axis. Since there are 32 non-terminal states in the Gridworld 3x3 domain, returning $32,000$ optimal actions is optimal. Averages are based on 10 and 30 repetitions for the experiments with and without auxiliary loss, respectively. 95%-confidence intervals are shown.

However, Figure 5.11 shows that there are three exceptions to the observation that introducing the auxiliary loss improves upon the generalization performance:

- **Generalization to large intervals.** Generalization to superfluous feature values sampled from very large intervals tends to be better for this domain if first-layer representations that are not entirely indifferent to the superfluous feature are closer to a Q^*-irrelevance abstraction. Notice that while introducing the auxiliary loss leads to improved generalization to small and moderately sized intervals, it deteriorates the generalization performance for large intervals (see Figure 5.11).

[10] Results for more hidden layer sizes and for the (OH)(N) state encoding are provided in Figure A.53 in the Appendix.

[11] The higher standard deviations for the experiments with auxiliary loss are largely due to the fact that 30 repetitions are performed when no auxiliary loss is used and solely 10 when the auxiliary loss is added due to time constraints.

This can be explained by the fact that even though the first-layer representations learn to ignore the superfluous feature to a larger extent when we apply the auxiliary loss, they do not do so completely. At the same time, the Euclidean distances between ground states with the same optimal action are on average clearly smaller than the ones between ground states with different optimal actions in a Q^*-irrelevance abstraction for Gridworld 3x3, but much less so in the coarsest Markov state representation[12]. Hence, that for very different superfluous feature values a state is mapped to an activation in the first layer that is near the activations of states with a different optimal action is more likely in this scenario than if the formed first-layer representation is closer to a Q^*-irrelevance abstraction, which is the case when no bisimulation-based auxiliary loss is used. This conclusion is also supported by the observation that 2-layer DQNs with a hidden layer size of 5, which form a first-layer representation more similar to a Q^*-irrelevance abstraction than DQNs with larger hidden layer sizes when the auxiliary loss is added[13], on average generalize better to very large intervals than DQNs with any other tested hidden layer size (see Figure 5.11a).

- **DQNs with large hidden layers.** Introducing the bisimulation-based auxiliary loss tends to lead to worse generalization performance to large intervals for large hidden layer sizes such as 65 than for smaller hidden layer sizes (see Figure 5.11). This is the case, because the first-layer representations of DQNs with large hidden layers become less similar to the coarsest Markov state representation again at the end of training for our hyperparameter settings for the auxiliary loss. Therefore, they begin to distinguish states based on the superfluous feature value to a higher degree (see Figure A.47b in the Appendix). Thus, while one would expect DQNs with varying hidden layer sizes to generalize similarly well to superfluous feature values unseen during training if the first-layer representations are very similar to the coarsest Markov state representation, this fact explains why this is not observed in our experiments.

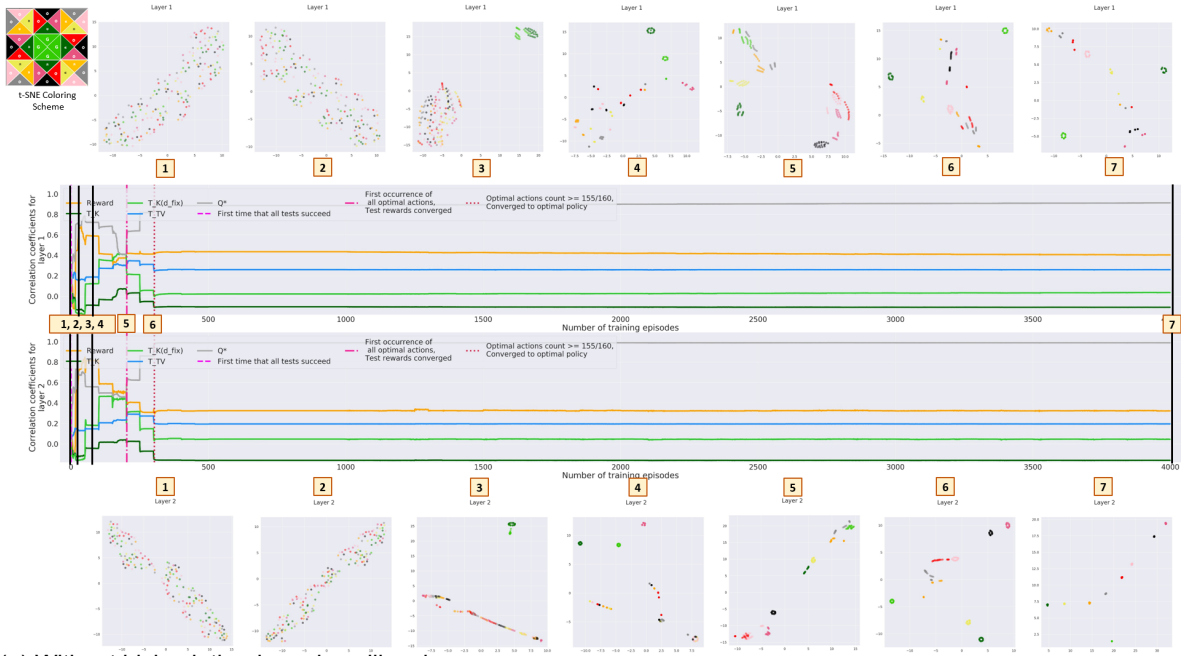
- **DQNs with very small hidden layers.** Besides the generalization performance of DQNs with large hidden layers to large intervals, there appears to be another exception to the observation that adding the auxiliary loss to the training improves upon the generalization performance. This exception are DQNs with a hidden layer size of 5, for which adding the auxiliary loss deteriorates generalization performance for superfluous feature values sampled from small intervals (see Figure 5.11a). The reason for this phenomenon is that applying the auxiliary loss to the training of DQNs with this hidden layer size does not guarantee that the networks always converge to the optimal policy by the end of training, even if very low weights are used for the auxiliary loss[14]. The relatively large standard deviations together with the fact that the average number of learned optimal actions is higher for large intervals when the auxiliary loss is added than when it is not used suggest, however, that if such a 2-layer DQN does converge to the optimal policy, the generalization is better when the auxiliary loss is added than when it is not introduced. This intuitively makes sense, because Figure 5.12 visualizes that utilizing the bisimulation-based auxiliary loss during the training of a 2-layer DQN with such a small hidden layer size does cause states that differ only in the superfluous feature value to be mapped to more similar activations in the first layer than if no auxiliary loss is utilized. Moreover, the final first-layer representation is at the same time rather close to a Q^*-irrelevance abstraction[15], which facilitates generalization to superfluous feature values sampled from very large intervals.

[12] Non-terminal ground states have average Euclidean distances of 0.175 and 0.333 to other non-terminal ground states with the same and different optimal actions, respectively, in a Q^*-irrelevance abstraction for Gridworld 3x3. In the coarsest Markov state representation, however, the mean Euclidean distances of non-terminal ground states to other non-terminal ground states with the same and different optimal actions are 0.141 and 0.144, respectively, if $d_E^{max} = d_B^{max} = 1$. The mean Euclidean distances are therefore much more similar for the coarsest Markov state representation than for a Q^*-irrelevance abstraction.

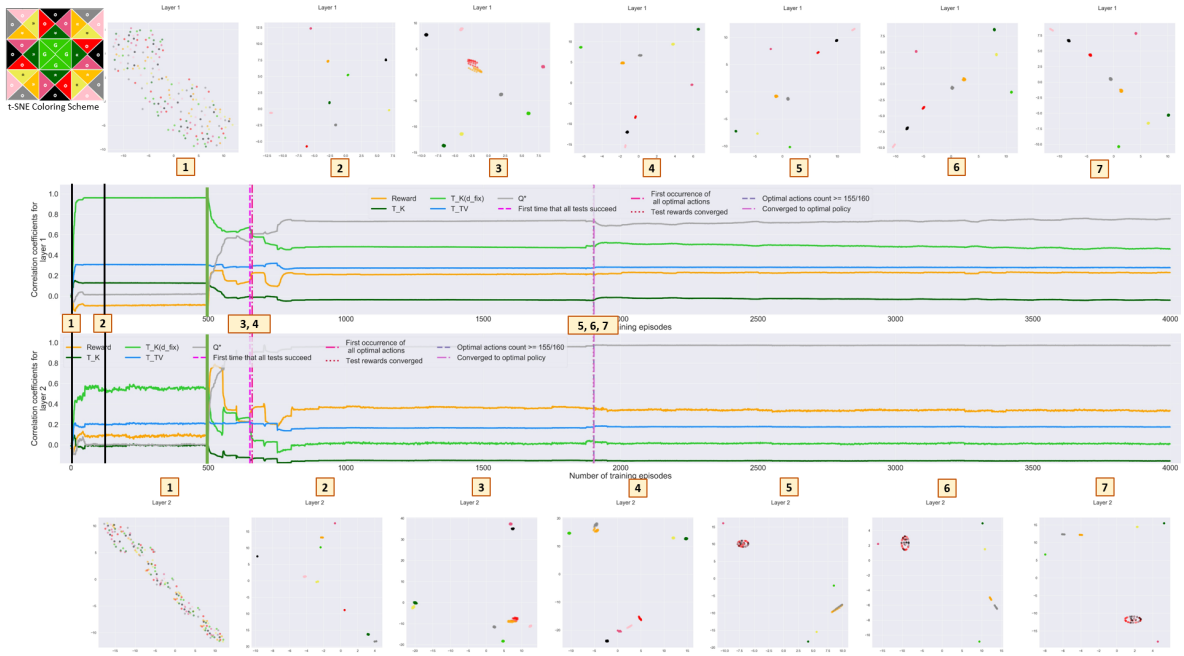
[13] This is due to the fact that the auxiliary loss is applied less intensively for DQNs with a hidden layer size of 5 in our experiments to not prevent the DQNs from learning the optimal policy. Refer to Section A.3.2.1 in the Appendix for more information on how we apply the auxiliary loss.

[14] Recall that DQNs with small hidden layers have to form a first-layer representation that is very similar to a Q^*-irrelevance abstraction to be able to learn accurate Q-values.

[15] This is due to the fact that the auxiliary loss is applied less intensively for DQNs with a hidden layer size of 5 to not prevent them from learning accurate Q-values. Refer to Section A.3.2.1 in the Appendix for more information on how we apply the auxiliary loss.



(a) Without bisimulation-based auxiliary loss.



(b) With bisimulation-based auxiliary loss.

Figure 5.12: Correlation coefficients and t-SNE plots of the activations states are mapped to in the layers of 2-layer DQNs with a hidden layer size of 3 for the Gridworld 3x3 (OH) domain during training. In a) the DQN is trained without auxiliary loss, and in b) the bisimulation-based auxiliary loss is applied based on d_{fix} during the first 500 training episodes with a weight of 0.1, a decay rate of 0.9999 and a value of $\sqrt{32 \times hidden\_layer\_size}$ for d_E^{max}.

6

Conclusion and Future Research

This chapter provides summarized answers to each of our three primary research questions in Section 6.1, and addresses the limitations of our approach and names directions for future work in Section 6.2.

6.1. Conclusion

Our work is based on the premise that a deep RL agent should ideally learn the coarsest Markov state representation, in which Euclidean distances between states are proportional to how behaviorally similar these states are. Behavioral similarity thereby is measured by a bisimulation metric that is based on the Kantorovich distance[1]. Based on this concept of ideal state representation, we examined what deep RL agents learn in practice. Moreover, we empirically tested whether creating this ideal representation in hidden layers is beneficial in the contexts of learning speed and generalization. In the sequel, we answer each of our main research questions:

1. **Which internal state representations do deep RL agents form during training and how similar are these to the coarsest Markov state representation?**
   Our experimental results suggest the existence of three overlapping learning phases in all network layers. First, states are clustered based on multi-step rewards, second, the internal state representations become more similar to the coarsest Markov state representation and especially more Markov with respect to the transition function, and finally, states are progressively grouped based on Q-values. Thus, learning phases 2 and 3 determine how similar to the coarsest Markov state representation the internal state representations become during and still are at the end of training. Thereby, the internal state representations formed in hidden layers during these two phases depend on the necessity, difficulty and feasibility of learning the coarsest Markov state representation and of grouping states based on Q-values. Factors that contribute to these three aspects are the network capacity and the state encoding. The precise coarsest Markov state representation is not learned during any of our experiments.

2a. **To which degree does creating internal state representations that are more similar to the coarsest Markov state representation *during* training improve upon the learning speed and consistency of deep RL agents?**
   Introducing a bisimulation-based auxiliary loss to push a 2-layer DQN to form an internal state representation in its hidden layer that is more similar to the coarsest Markov state representation can cause a DQN to converge to the true Q-values more quickly and reliably. The best results are thereby obtained when the auxiliary loss forces a DQN to create the *precise* coarsest Markov state representation. In that case, the auxiliary loss is based on an exact bisimulation metric that is costly to compute in practice. If we replace this exact bisimulation metric by an approximate bisimulation metric or a component of bisimulation metrics, a DQN is pushed to create an *approximate* coarsest Markov state representation or a representation that is *no coarser* than the

[1] The Kantorovich distance is also referred to as Wasserstein distance, Kantorovich-Rubinstein distance, Monge-Kantorovich distance or earth mover's distance.

coarsest Markov state representation, respectively. In that case, learning proceeds also more quickly at the beginning of training, but no faster convergence to the true Q-values is ultimately achieved. While it is thus possible to remove the bottleneck of having to compute the exact bisimulation metric required for learning the coarsest Markov state representation, doing so comes at the expense of decreased effectiveness.

2b. **To which extent does learning internal state representations that are more similar to the coarsest Markov state representation *by the end of* training lead to improved generalization?**
Generalization to modified reward and transition functions. We pretrained 2-layer DQNs on an original domain and subsequently retrained them on related domains with modified reward or transition functions and as relevant features a subset of the relevant features of the original domain. We find that if the internal state representation transferred from the first layer of a pretrained DQN is more similar to the coarsest Markov state representation, earlier convergence and convergence to more accurate Q-values are achieved on a related domain. Yet, the usefulness of learning a first-layer representation that is similar to the coarsest Markov state representation depends on the specific changes made to the reward and transition functions. Specifically, generalization based on a first-layer representation that is less similar to the coarsest Markov state representation tends to be worse for related domains with certain modifications to the transition function. These are modifications that necessitate a precise representation of the transition function to learn the new Q-values, but overall lead to relatively low differences in Q-values compared to the original domain. Finally, adding an auxiliary loss that pushes a network to form the coarsest Markov state representation in its hidden layer improves upon the generalization performance for settings for which the generalization after pretraining without auxiliary loss is poor.

Generalization to new irrelevant feature values. Internal first-layer state representations that are more similar to the coarsest Markov state representation cause better generalization to superfluous feature values that are *moderately* different from the ones encountered during training. Yet, better generalization to superfluous feature values *very* different from the ones seen during training is not always attained if the learned first-layer internal state representation is more similar to the coarsest Markov state representation. Specifically, if the created first-layer representation does not entirely ignore a superfluous feature, generalization to such values is better if the Euclidean distances between states with different optimal actions are larger relative to those of states with the same optimal action than they are in the coarsest Markov state representation.

6.2. Directions for Future Research

We subsequently outline suggestions for future work with respect to scaling up, generalization, and other learning algorithms.

6.2.1. Scaling Up

One limitation of our experiments is that they are based on small domains and require computations which in their original form are not scalable to larger problems. We therefore identify the following directions for future work:

- *Domains*. The largest one of our fully observable domains is the Gridworld 5x5 domain, which has a state space size of only 500 after introducing the superfluous feature. While using such small domains has the advantage that we can identify each individual state in t-SNE plots to get a clear understanding of what a deep RL agent has learned at any point during training, it is desirable to see if our observations also hold for more realistic domains. Furthermore, while we do explore the internal state representations formed by a DRQN for the partially observable Hallway domain, this domain also merely has 15 ground states and we do not perform any generalization experiments for partially observable domains. Hence, more experiments should be conducted both for simple and more realistic partially observable domains.
- *Computation of bisimulation metrics for stochastic domains*. The only somewhat efficient algorithm to compute the Kantorovich distance-based bisimulation metric for continuous MDPs with stochastic transitions is the Monte Carlo approach by [13], yet it is not scalable to very large domains. Furthermore, while the approximation algorithm by [8] is applicable to large domains, it

is not designed for domains with stochastic transitions. Thus, the bisimulation metric which we ideally want Euclidean distances between states to be proportional to cannot easily be computed for large domains with stochastic transitions. This limits the applicability of our auxiliary loss in its current form and of our bisimulation-based correlation coefficients. Future work should consequently find a way to more efficiently approximate this bisimulation metric also for stochastic domains.

- *Incorporate computation of bisimulation metrics into learning*. Our auxiliary loss is not only limited in applicability because the Kantorovich distance-based bisimulation metric cannot be efficiently computed for stochastic domains, but also because calculating bisimulation metrics for the auxiliary loss and learning itself currently are separate, consecutive steps. Yet, these steps could potentially be combined. For example, since the approximation algorithm by [8] also computes encodings of observations to calculate the bisimulation metric, it would be useful to investigate how to directly incorporate the approximation of the bisimulation metric into the training of a DQN. Notice that our experiments in Chapter 4 reveal that we can replace the expensive computation of the precise Kantorovich distance-based bisimulation metric by an approximation by means of the algorithm by [8], albeit at the expense of reduced effectiveness of the resulting auxiliary loss with respect to learning speed. Yet, using this approximation algorithm introduces additional steps, such as the tuning of hyperparameters and the training of the neural network used for the approximation. While the training time required for the small domains utilized in our work is negligible as it is in the order of seconds[2], more time is likely needed for realistic domains.

- *On-policy bisimulation metrics*. When starting our research, we had identified using on-policy bisimulation metrics as a way to in the future make our auxiliary loss more scalable to larger domains. Such on-policy bisimulation metrics do not take *all actions* into consideration for computing behavioral similarity, but only those of a specific *policy*. While the concurrent research by [64] has already explored the use of an auxiliary loss based on on-policy bisimulation metrics, it would be insightful to directly compare the impacts of utilizing on-policy bisimulation metrics rather than the original bisimulation metrics, especially with respect to generalization. Note that a representation learned based on on-policy bisimulation metrics is no finer than one formed based on bisimulation metrics, as solely a subset of actions is considered in on-policy bisimulation metrics.

6.2.2. Generalization

We empirically examined the importance of learning an internal state representation that is similar to the coarsest Markov state representation for generalization in Chapter 5. We propose to explore the following aspects in future research:

- We found in Chapter 5 that certain modifications to the transition function make it more important to have learned an internal first-layer state representation that is similar to the coarsest Markov state representation. It would thus be interesting to investigate in more detail the impacts of different types of modifications on the usefulness of learning the coarsest Markov state representation.

- The work of [30] examines the impact of different forms of non-stationarity on the generalization performance of neural networks. To gain insights into the effect a type of non-stationarity has on the formed internal state representations, it would be worthwhile to compute our bisimulation-based correlation coefficients.

6.2.3. Other Learning Algorithms

We utilize only DQNs and DRQNs with hard target network updates for our experiments. The impact of this is very visible from the correlation coefficients during training and also the kinds of internal state representations that are present at certain stages of training. It would hence be interesting to look at the impact of applying soft target network updates and of employing other learning algorithms on the types of internal state representations that are learned. At the same time, computing our bisimulation-based correlation coefficients for other network architectures and learning algorithms could help to gain a deeper understanding of what the resulting deep RL agents learn.

[2]Refer to Section D.3.1 in the Appendix for information on how we run our computations.

Bibliography

[1] Joshua Achiam, Ethan Knight, and Pieter Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

[2] Frangioni Antonio and Antonio Manca. A computational study of cost reoptimization for min cost flow problems. *INFORMS Journal on Computing*, 18, 04 2003. doi: 10.1287/ijoc.1040.0081.

[3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[4] Giorgio Bacci, Giovanni Bacci, Kim G. Larsen, and Radu Mardare. On-the-fly exact computation of bisimilarity distances. In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'13, page 1–15, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 9783642367410. doi: 10.1007/978-3-642-36742-7_1. URL https://doi.org/10.1007/978-3-642-36742-7_1.

[5] Giorgio Bacci, Giovanni Bacci, Kim G Larsen, and Radu Mardare. Computing behavioral distances, compositionally. In *International Symposium on Mathematical Foundations of Computer Science*, pages 74–85. Springer, 2013.

[6] Francesco Bertoluzzo and Marco Corazza. Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance*, 3:68–77, 2012.

[7] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

[8] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. *arXiv preprint arXiv:1911.09291*, 2019.

[9] Gheorghe Comanici, Prakash Panangaden, and Doina Precup. On-the-fly algorithms for bisimulation metrics. *2012 Ninth International Conference on Quantitative Evaluation of Systems*, pages 94–103, 2012.

[10] Sven F Crone, Stefan Lessmann, and Robert Stahlbock. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Journal of Operational Research*, 173(3):781–800, 2006.

[11] Norm Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *UAI*, 2014.

[12] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 162–169. AUAI Press, 2004.

[13] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[14] Norman Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in markov decision processes. *arXiv preprint arXiv:1206.6836*, 2012.

[15] Norman Ferns, Prakash Panangaden, and Doina Precup. Metrics for markov decision processes with infinite state spaces. *arXiv preprint arXiv:1207.1386*, 2012.

[16] Elena Fitkov-Norris, Samireh Vahid, and Chris Hand. Evaluating the impact of categorical data encoding and scaling on neural network classification performance: The case of repeat consumption of identical cultural goods. In *EANN*, 2012.

[17] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3582–3589, 2019.

[18] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends in Machine Learning*, 11: 219–354, 2018.

[19] Artur d'Avila Garcez, Tarek R Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C Lamb, Risto Miikkulainen, and Daniel L Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *2015 AAAI Spring Symposium Series*, 2015.

[20] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *ArXiv*, abs/1906.02736, 2019.

[21] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801, 2018.

[24] Carlos Guestrin, Daphne Koller, and Ronald Parr. Solving factored pomdps with linear value functions. In *IJCAI 2001*, 2001.

[25] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[26] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[27] Tiew Kee Hui. Reinforcement learning: Q-learning with open ai taxi, 2019. `https://tiewkh.github.io/blog/qlearning-openaitaxi/`.

[28] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9 (3):90, 2007.

[29] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, pages 13978–13990, 2019.

[30] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

[31] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[32] Svante Janson. Probability distances, 2020. `http://www2.math.uu.se/~svante/papers/sjN21.pdf`.

[33] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[35] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[36] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[37] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[38] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 1995. URL `http://people.csail.mit.edu/lpk/papers/ml95.ps`.

[39] Michael L Littman, Thomas L Dean, and Leslie Pack Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 394–402. Morgan Kaufmann Publishers Inc., 1995.

[40] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[41] Sridhar Mahadevan. Representation discovery in sequential decision making. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[42] R McCallum. Reinforcement learning with selective perception and hidden state. 1997.

[43] Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 427–436. Morgan Kaufmann Publishers Inc., 1999.

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

[45] Andrew Y Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger, and Eric Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental robotics IX*, pages 363–372. Springer, 2006.

[46] Adam Paszke. Reinforcement learning (dqn) tutorial. `https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html`.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[49] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[50] Kedar Potdar, Taher S. Pardawala, and Chinmay D. Pai. A comparative study of categorical variable encoding techniques for neural network classifiers. 2017.

[51] Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19:47, 2015.

[52] Gavin A Rummery and Mahesan Niranjan. *Online Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[53] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

[54] John W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18:401–409, 1969.

[55] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4967–4976. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7082-a-simple-neural-network-module-for-relational-reasoning.pdf.

[56] Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *ICML*, volume 2, page 6, 2011.

[57] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[58] Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[59] Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and M. Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *AAMAS*, 2020.

[60] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[61] Tiago Veiga, Matthijs T. J. Spaan, and Pedro U. Lima. Point-based pomdp solving with factored value function approximation. In *AAAI*, 2014.

[62] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989. URL http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf.

[63] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016. doi: 10.23915/distill.00002. URL http://distill.pub/2016/misread-tsne.

[64] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[65] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.

[66] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017. URL https://arxiv.org/abs/1611.01578.

A

Detailed Results

This chapter contains additional results and figures for the experiments discussed in the main part of this work for each of our three primary research questions. Specifically, Sections A.1, A.2 and A.3 supply further information on the characteristics of internal state representations during learning, the impact of Markovianity on learning speed and consistency, and the impact of Markovianity on generalization, respectively.

A.1. Characteristics of Internal State Representations During Learning

Section A.1.1 shows supporting results for the second learning phase of DQNs and Section A.1.2 further information on the factors impacting the learning process. In addition, Section A.1.3 discusses whether the bisimulation-based correlation coefficients can be utilized to infer whether a DQN has discovered or converged to the optimal policy, Section A.1.4 explores whether bisimulation-based correlation coefficients enable drawing conclusions regarding the adequacy of a DQN's capacity, and Section A.1.5 supplies further figures.

A.1.1. Learning Phase 2

We provide experimental results for three more domains besides the Gridworld 3x3 domain in Section A.1.1.1. In addition, we show that the hidden- and output-layer state representations also become more similar to the coarsest Markov state representation during this phase of learning in a domain in which states with the same Q-values are not necessarily bisimilar in Section A.1.1.2, and when a fixed replay memory is used during training in Section A.1.1.3. Furthermore, the first-layer representation tends to be more similar to the coarsest Markov state representation and more Markov with respect to the transition function than the ones in subsequent hidden layers as discussed in Section A.1.1.4. Lastly, we delineate that both $c_{K(d_{fix})}$ and $c_{d'_{fix}}$, the latter of which is much faster to compute, enable insights into how similar to the coarsest Markov state representation an internal state representation is in Section A.1.1.5.

A.1.1.1 Further Domains

Gridworld 5x5 and FrozenLake 8x8. Figures A.1 and A.2 visualize that the hidden- and output-layer representations also become more similar to the coarsest Markov state representation and more Markov with respect to the transition function as states are grouped based on multi-step rewards for the Gridworld 5x5 and FrozenLake 8x8 domains. Furthermore, Figure A.3 mirrors that it also holds for a 2-layer DQN for the FrozenLake 8x8 (OH)(F-OH) domain that both c_{TV} and $c_{K(d_{fix})}$ tend to be higher in the first than in the input[1] and output layers.

[1] Recall that the term *input layer* refers to the state encoding.

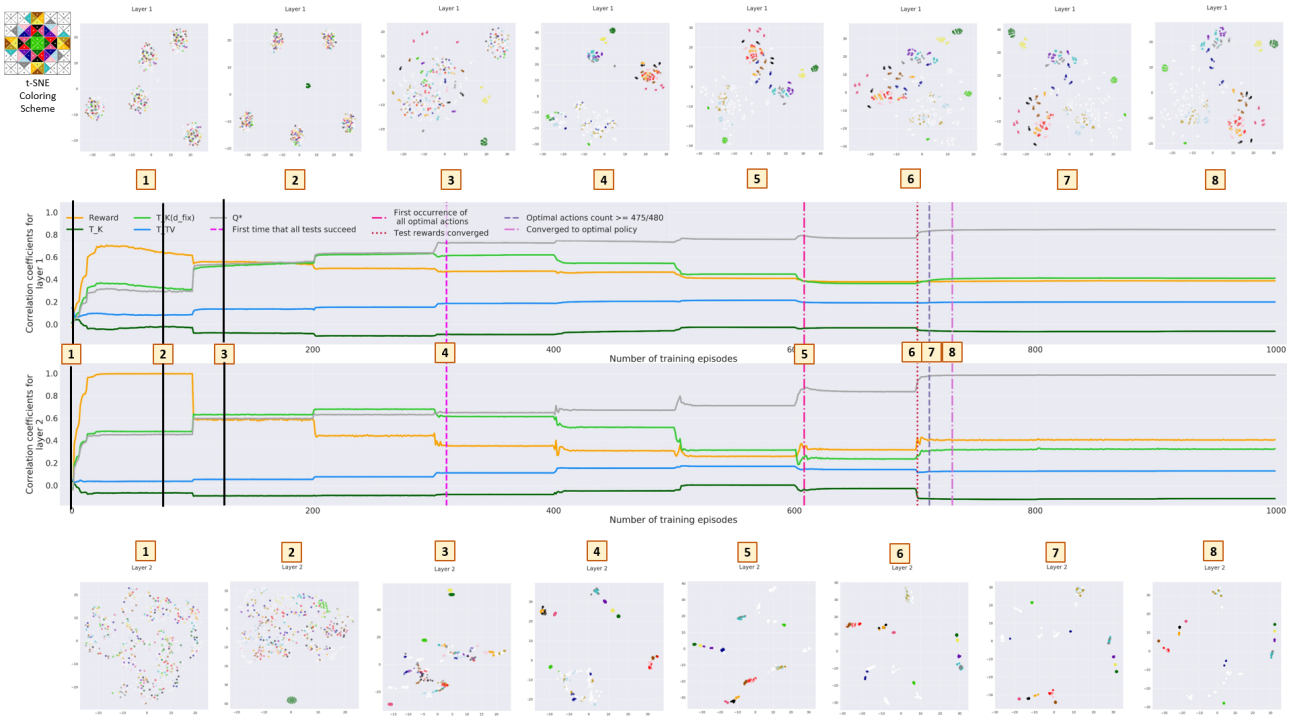


Figure A.1: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the Gridworld 5x5 (OH)(F-OH) domain. The hidden layer size is equal to 50.

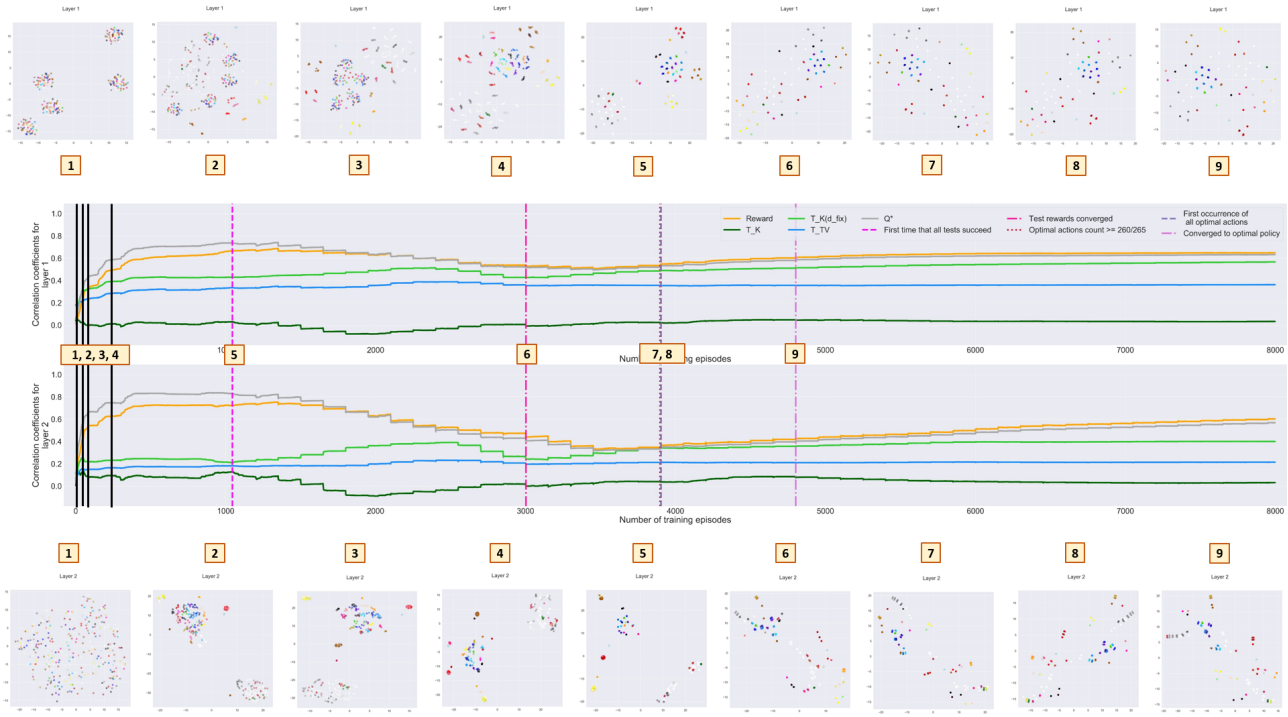


Figure A.2: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the FrozenLake 8x8 (OH)(F-OH) domain. The hidden layer size is equal to 50.

FrozenLake 4x4. Clustering states based on immediate rewards for this domain already leads to almost the same clusters of states as when grouping states based on Q-values. Moreover, c_{TV} and $c_{K(d_{fix})}$ are rather high for a Q^*-irrelevance abstraction for the FrozenLake 4x4 domain. Consequently, it is not visible for this domain when looking at t-SNE plots of the first-layer representation and the values for c_{TV} and $c_{K(d_{fix})}$ that the internal state representations learn to represent the transition function and

Figure A.3: Mean c_{TV} and $c_{K(d_{fix})}$ for the layers of 2-layer DQNs for the FrozenLake 8x8 (OH)(F-OH) domain. Values are based on 5 repetitions and 95%-confidence intervals are shown. The hidden layer size is equal to 12.

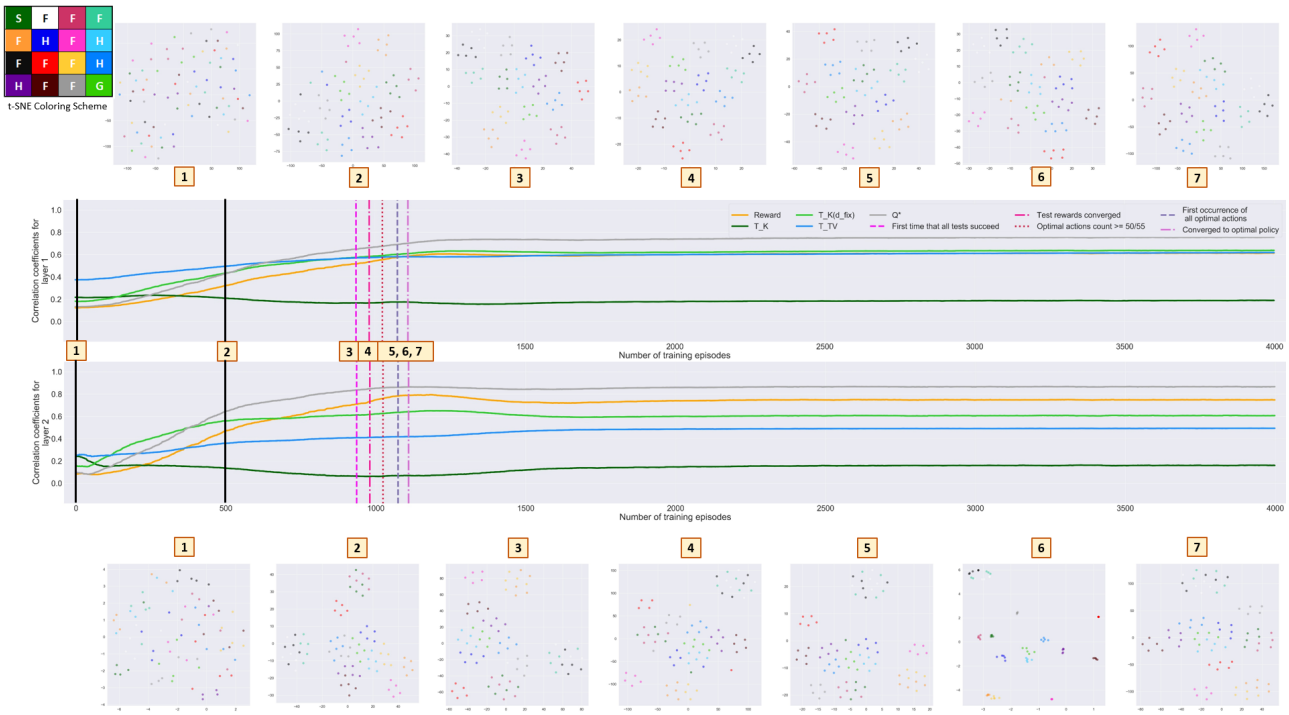


Figure A.4: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the FrozenLake 4x4 (OH)(F-OH) domain. The hidden layer size is equal to 50.

become closer to the coarsest Markov state representation to some extent, rather than just forming close to a Q^*-irrelevance abstraction, during this learning phase (see Figure A.4).

What can nicely be seen for the FrozenLake 4x4 domain is that the higher values of c_{TV} and $c_{K(d_{fix})}$ in the first network layer compared to the input and output layers largely stem from different distances between the abstract states that are created as well as from forming close but clearly distinct clusters for non-bisimilar states with similar Q-values. For example, the states whose activations are drawn in black, blue-green and white in t-SNE plots have similar Q-values, as they are all located above a hole and require similar numbers of steps to the goal under the optimal policy. However, the three corresponding ground states are not bisimilar. The activations that these states are mapped to tend to form a single cluster in the second layer of a DQN for FrozenLake 4x4 (OH)(F-OH), whereas they are separated in three clearly distinct clusters in the first layer (see Figure A.4).

A.1.1.2 Gridworld 3x3 (Aug)

The hidden- and output-layer representations become more similar to the coarsest Markov state representation at the beginning of training even for a domain in which states with the same Q-values are not necessarily bisimilar. Recall that in the Gridworld 3x3, Gridworld 5x5, FrozenLake 4x4 and FrozenLake 8x8 domains, states have the same Q-values if and only if they are bisimilar. Thus, to rule out that the internal state representations become more similar to the coarsest Markov state representation at

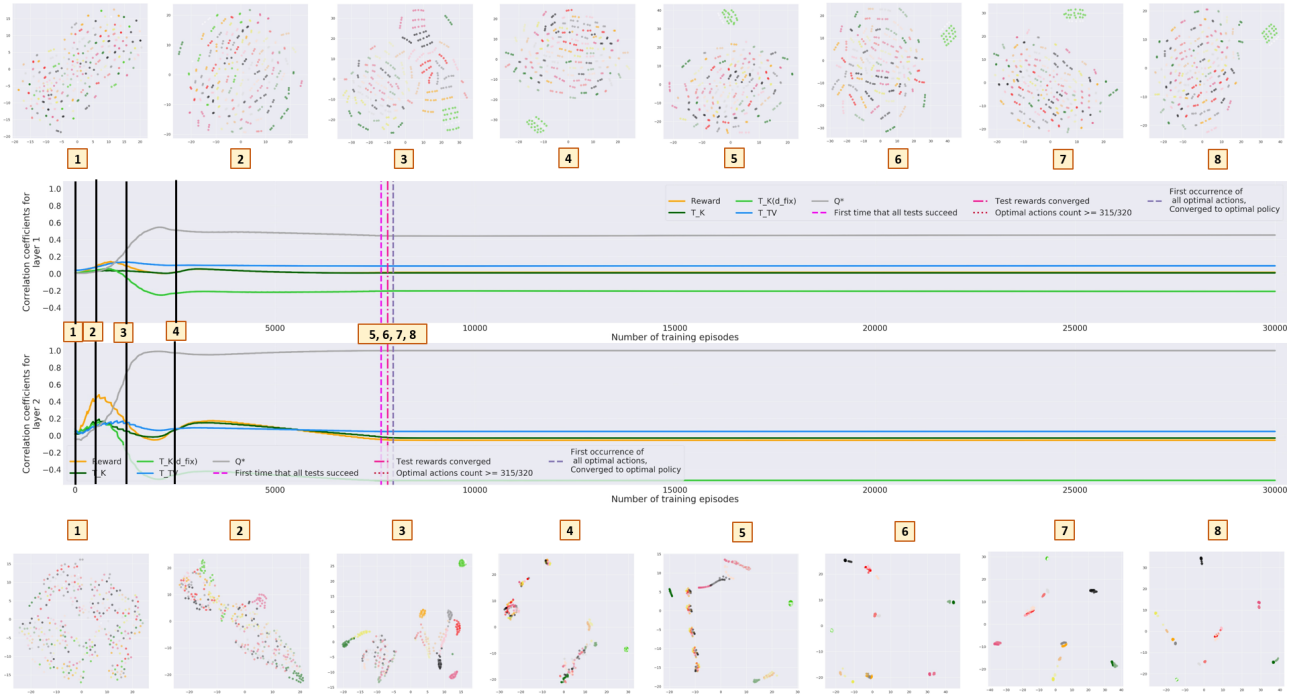


Figure A.5: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the Gridworld 3x3 (Aug) (OH) domain. The hidden layer size is equal to 50.

the beginning of training only because the abstract states of the coarsest Markov state representation and a Q^*-irrelevance abstraction are the same, we also conducted experiments with the Gridworld 3x3 (Aug) domain, in which states with the same Q-values are not necessarily bisimilar. Figure A.5 shows that bisimilar states rather than states with the same Q-values are at first grouped together. Notice that $c_{K(d_{fix})}$ initially increases, whereas c_{Q^*} stays the same or even decreases. Furthermore, in t-SNE plots 3, only states whose activations are drawn in exactly the same color and not states whose activations are colored in paler and darker shades of the same color tend to be mapped to similar activations[2]. This likely is the case, because it is necessary to have learned rather precise estimates of the true Q-values for this domain to realize that some non-bisimilar states have the same Q-values, which has not yet been achieved during this phase of learning.

A.1.1.3 Impact of Exploration

To rule out that the observed way in which the internal state representations are formed arises solely due to the way in which an agent increasingly explores a domain, we also conducted experiments in which transitions during training are sampled from a fixed replay memory. This replay memory is previously filled during the training of an agent on the same domain. Since the replay memory size is larger than the total number of transitions performed during training, the replay memory contains all transitions encountered during training of a DQN without fixed replay memory. Figure A.6 shows for a 2-layer DQN for the Gridworld 3x3 (Aug) (OH)(N) domain that the internal state representations become more similar to the coarsest Markov state representation at the beginning of training even when such a fixed replay memory is utilized[3]. This can be seen from the initially increasing value of $c_{K(d_{fix})}$ and the fact that c_{Q^*} decreases at the beginning of training. In addition, especially t-SNE plots 4 clearly depict that bisimilar states rather than states with the same Q-values are mapped to similar activations.

Moreover, while c_{TV} decreases at the beginning of training in Figure A.6, this also occurs for the (OH)(N) encoding for this domain when no fixed replay memory is utilized (see Figure A.27). This is due to the high input-layer and initial first-layer value for c_{TV} for this form of state encoding. More precisely, all states differing solely in the superfluous feature value are already mapped to very similar activations

[2]States whose activations are drawn in different shades of the same color have the same Q-values but are not bisimilar. Refer to Section 3.1.5 for more information on the t-SNE coloring scheme.

[3]See Figure C.4 in the Appendix for the contents of the used fixed replay memory.

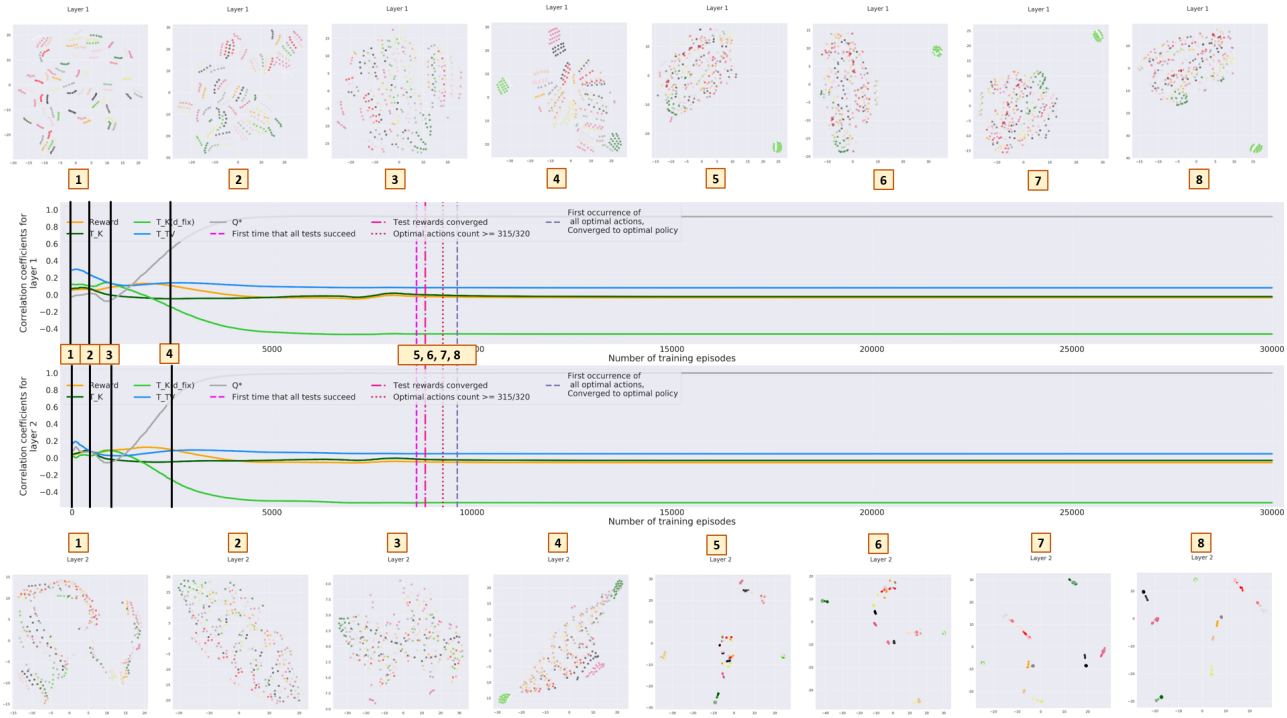
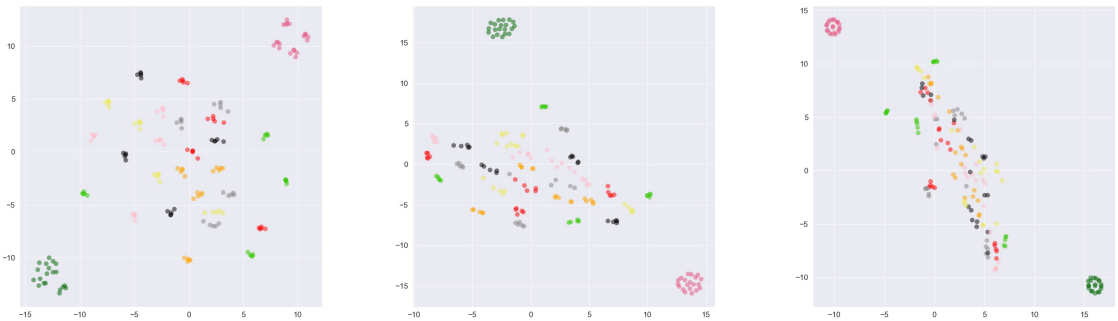


Figure A.6: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the Gridworld 3x3 (Aug) (OH)(N) domain when a fixed replay memory is utilized during training. The hidden layer size is equal to 6.

in the input layer and in the first layer at the start of training, and distances between the activations of states from different ground states are very alike. Consequently, c_{TV} cannot increase due to mapping states from the same ground state closer together, and it decreases as distances between the activations of states from different ground states become less similar during training. Further experiments with fixed replay memories for the Gridworld 3x3 and FrozenLake 4x4 domains that underline the conclusion presented here are discussed in Section C.1 in the Appendix.



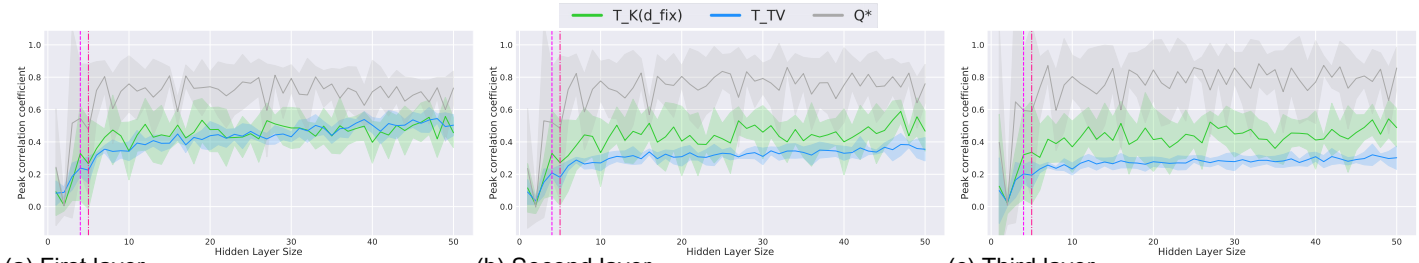
(a) First layer. (b) Second layer. (c) Third layer.

Figure A.7: t-SNE plots of the activations states are mapped to in the hidden layers of a 4-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH)(F-OH) domain after training episode 75. At this point during training, the target network has been updated once.

A.1.1.4 Impact of Multiple Hidden Layers

For 4-layer DQNs, the first-layer representation also generally becomes more similar to the coarsest Markov state representation and more Markov with respect to the transition function during this phase of training than the representations in later hidden layers. For instance, one can see in Figure A.7 that the first layer of a 4-layer DQN with a hidden layer size of 50 already forms separate clusters for all states differing solely in the superfluous feature value after the target network has been updated solely

once, whereas this occurs to lesser degrees for the second and third layers[4]. In fact, Figure A.8 shows that the mean peak values of $c_{K(d_{fix})}$ and c_{TV} in the first layers of 4-layer DQNs for the Gridworld 3x3 (OH)(F-OH) domain tend to be higher than in the second and the third layer for all sufficiently large hidden layer sizes.



(a) First layer. (b) Second layer. (c) Third layer.

Figure A.8: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in each hidden layer with 95%-confidence intervals for each hidden layer size for a 4-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer size for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

A.1.1.5 $c_{K(d_{fix})}$ vs. $c_{d'_{fix}}$

Since Euclidean distances in the coarsest Markov state representation are proportional to d_{fix}, $c_{K(d_{fix})}$ is a precise measure of how close to the coarsest Markov state representation an internal state representation is. However, the feasibility of utilizing $c_{K(d_{fix})}$ in practice is strongly limited by the long time needed for computing d_{fix} with a satisfying precision. Yet, as depicted in Figure C.6 in the Appendix and analyzed in more detail in Section C.3 in the Appendix, computing $c_{d'_{fix}}$ rather than $c_{K(d_{fix})}$ leads to very similar results for the FrozenLake 4x4 (OH) domain. This is promising, because since d'_{fix} can be calculated much faster than d_{fix}, an analysis of how close to the coarsest Markov state representation an internal state representation is based on $c_{d'_{fix}}$ is much more scalable to larger domains.

A.1.2. Factors Impacting Hidden-layer State Representations During Learning Phases 2 and 3

In the following, we provide supporting evidence for our analysis of the factors impacting the hidden-layer state representations during the second and third learning phases from Section 3.2.2.3.

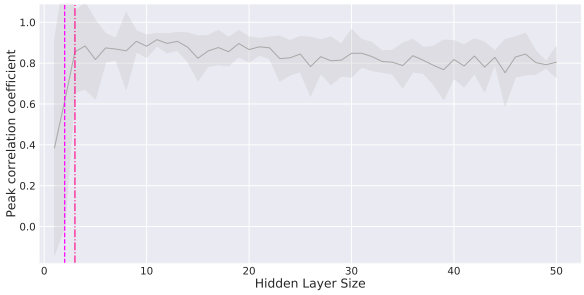
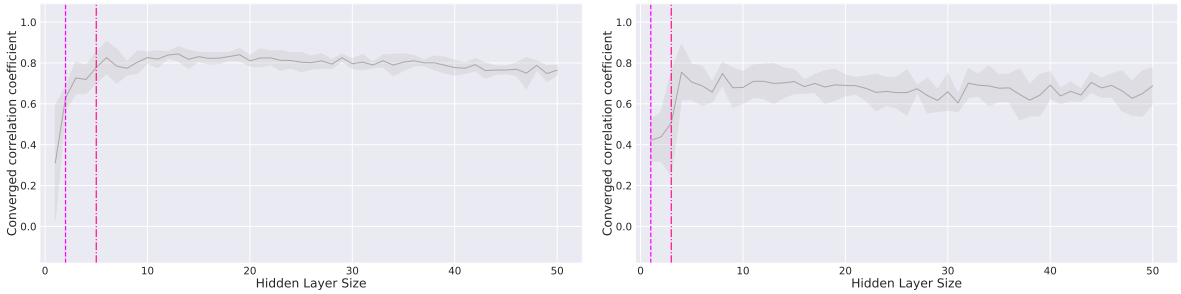


Figure A.9: Mean peak c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the Gridworld 5x5 (OH)(F-OH) domain. Values are based on 4 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 4 times.

Necessity. Figures A.9 and A.10 show that less close to a Q^*-irrelevance abstraction is formed in the first layer for larger-than-necessary than for just-right hidden layer sizes also for the Gridworld 5x5 and

[4]Recall that the activations of all bisimilar states are drawn in the same color for the Gridworld 3x3 domain. Thereby, always 4 ground states are bisimilar, and there are 5 states corresponding to each ground state.

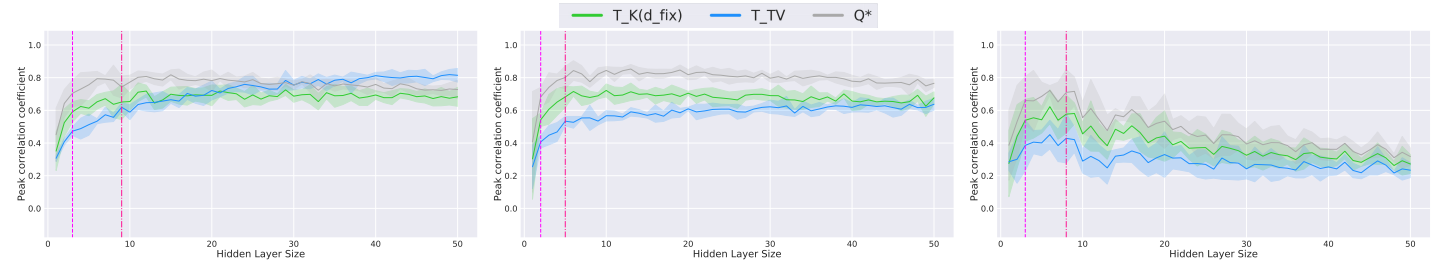
the FrozenLake 4x4 and FrozenLake 8x8 domains when the (OH)(F-OH) state encoding is used[5].



(a) FrozenLake 4x4. (b) FrozenLake 8x8.

Figure A.10: Mean converged c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the FrozenLake 4x4 (OH)(F-OH) and FrozenLake 8x8 (OH)(F-OH) domains. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

Difficulty. Our conclusions regarding the difficulty of forming internal state representations similar to the coarsest Markov state representation or to a Q^*-irrelevance abstraction are supported by results for the FrozenLake 4x4 domain. More precisely, Figure A.11 depicts that the degree to which the first-layer representation becomes similar to the coarsest Markov state representation, Markov with respect to the transition function, and close to a Q^*-irrelevance abstraction is contingent on the state encoding also for the FrozenLake 4x4 domain[6]. Just as for the Gridworld 3x3 domain, this difference is due to the varying input-layer and initial first-layer representations, which determine how difficult it is to map states differing solely in the superfluous feature value to similar activations (see Figure A.12).

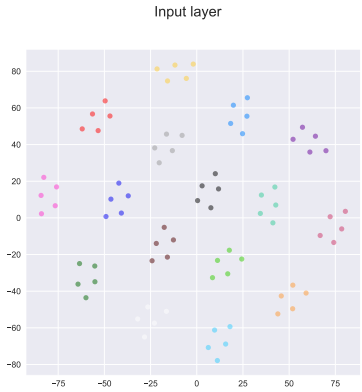


(a) FrozenLake 4x4 (OH)(N). (b) FrozenLake 4x4 (OH)(F-OH). (c) FrozenLake 4x4 (OH).

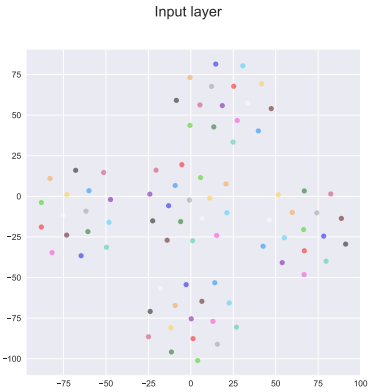
Figure A.11: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the FrozenLake 4x4 domain with different forms of state encoding. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

[5] Since the highest value for c_{Q^*} occurs not necessarily at the end of training for the FrozenLake 8x8 domain, we depict the mean converged value for c_{Q^*} for the FrozenLake domains.

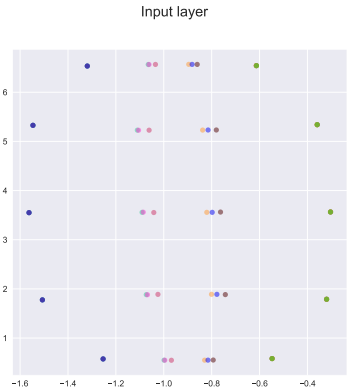
[6] Just as for Gridworld 3x3, the peak first-layer value for c_{TV} occurs at the start of training rather than during the second learning phase for the (OH)(N) state encoding for hidden layer sizes greater than or equal to the dimensionality of the state encoding (see Figures A.25a and A.26a).



FrozenLake 4x4 (OH)(N).

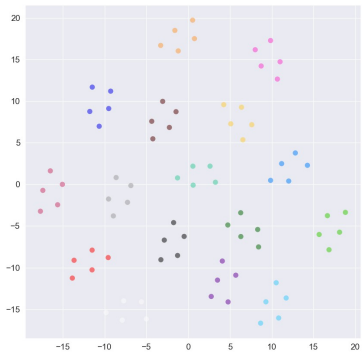


FrozenLake 4x4 (OH)(F-OH).

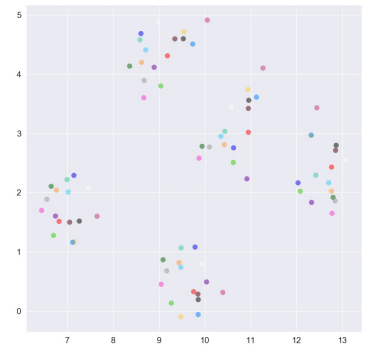


FrozenLake 4x4 (OH).

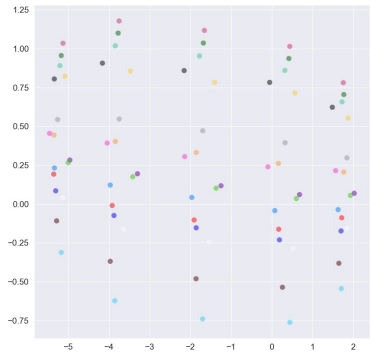
(a) Input Layer.



FrozenLake 4x4 (OH)(N).



FrozenLake 4x4 (OH)(F-OH).



FrozenLake 4x4 (OH).

(b) First Layer.

Figure A.12: t-SNE plots of the activations at the beginning of training for the input and first layers of 2-layer DQNs with a hidden layer size of 50 for the FrozenLake 4x4 domain with different forms of state encoding. Activations are computed for all states and every 5 states that differ solely in the superfluous feature and hence correspond to the same ground state are drawn in the same color.

Feasibility. Below, we provide further feasibility results for another state encoding in which the ground state is encoded via features for Gridworld 3x3 and for FrozenLake 4x4:

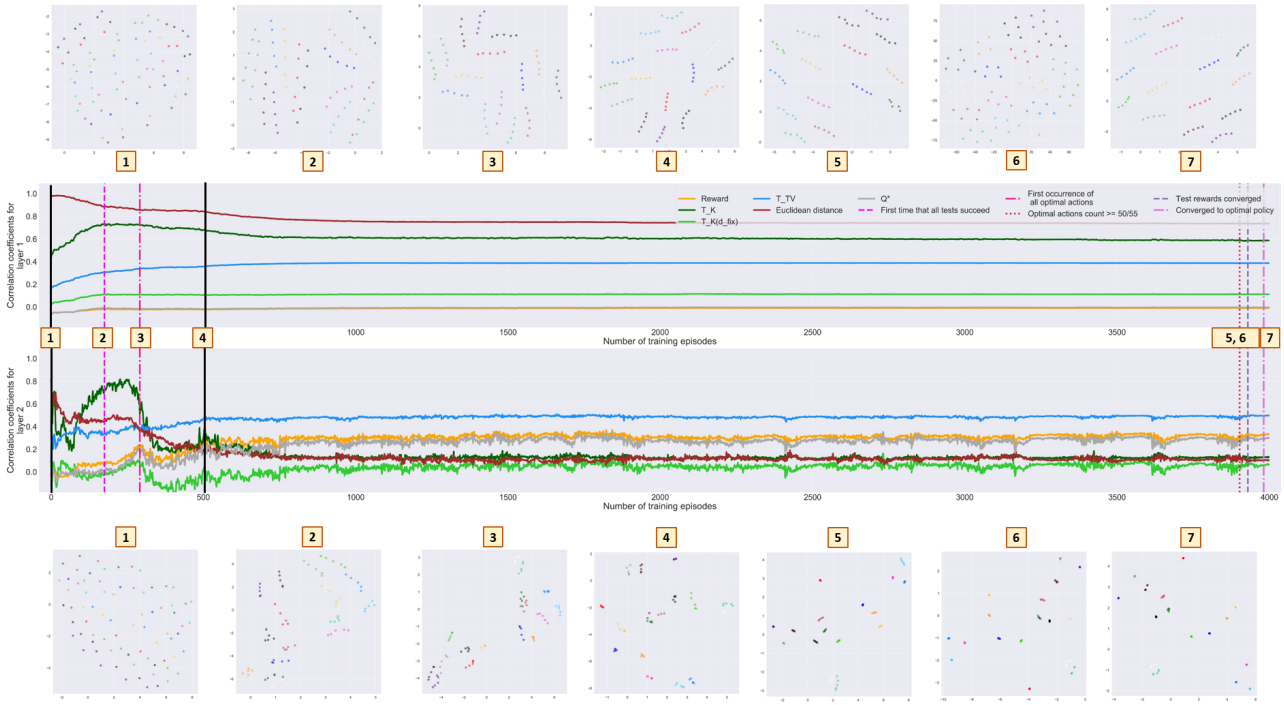
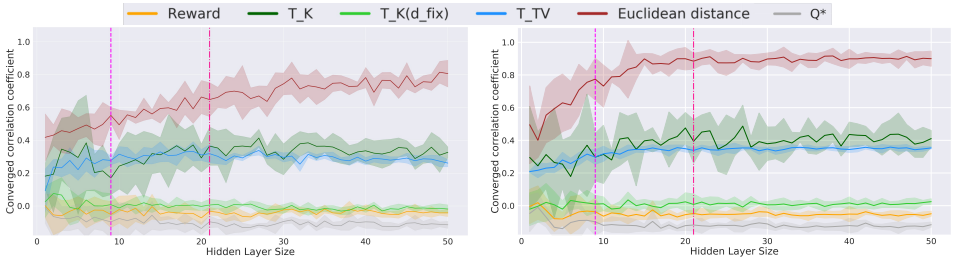


Figure A.13: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (F)(N) domain.

- **Gridworld 3x3**. As visualized in Figure A.13, the created first-layer representation also clusters the activations of states based on their ground states for the (F)(N) state encoding, for which all features are scaled to $[0, 1]$ and hence have the same scale. Yet, Figure A.14 shows that the first-layer value for c_E typically is higher at the end of training for the (F)(N) than for the (F) encoding. The reason for this is that the first layer learns to map states differing solely in the superfluous feature value to very similar activations, and that states that differ only in the superfluous feature value already have encodings with relatively small Euclidean distances when all features are scaled to $[0, 1]$.



(a) Gridworld 3x3 (F). (b) Gridworld 3x3 (F)(N).

Figure A.14: Mean converged c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV}, c_E and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the Gridworld 3x3 (F) and Gridworld 3x3 (F)(N) domains. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

- **FrozenLake 4x4**. As depicted in Figure A.15, the first-layer representation of a 2-layer DQN also learns to map states to similar activations if and only if they belong to the same ground state for the FrozenLake 4x4 domain when the ground state is encoded via features. Thereby, even a hidden layer size of 50 does not allow a 2-layer DQN for the FrozenLake 4x4 domain to converge to the true Q-values when the ground state is encoded via features. This is mirrored in Figure

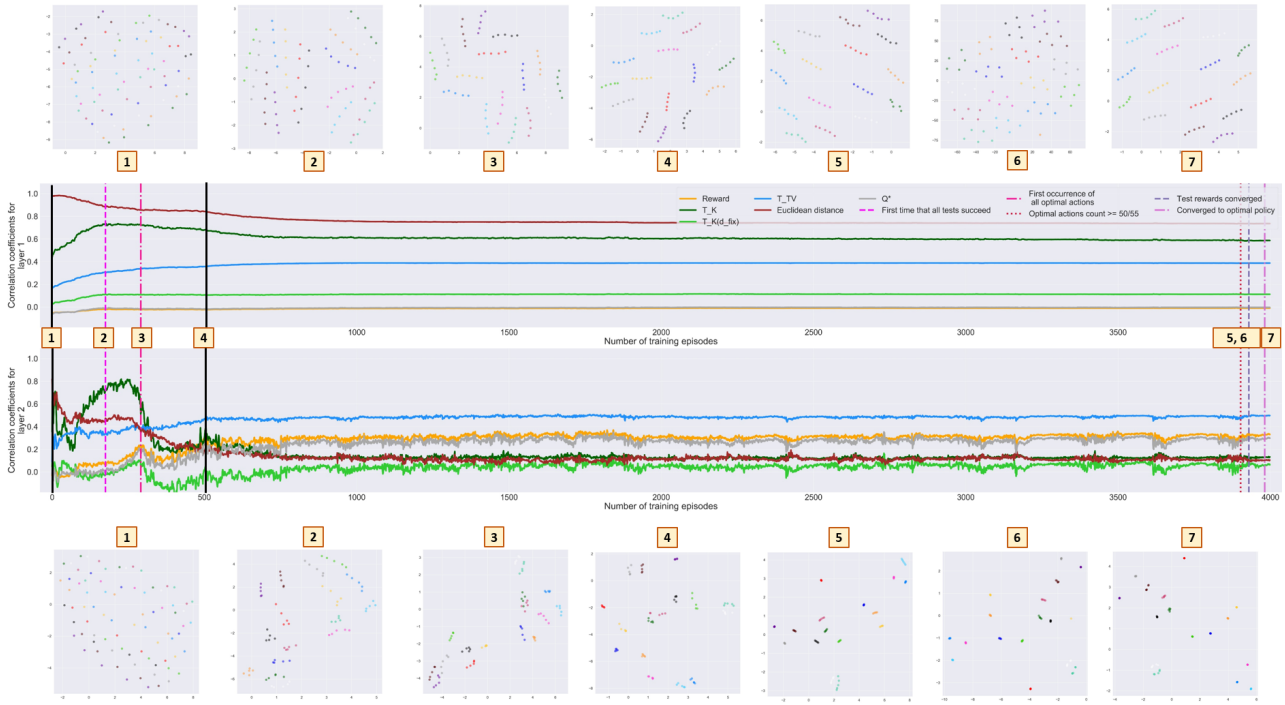


Figure A.15: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (F) domain.

A.16 by the fact that c_{Q^*} is not close to 1 for the output layer at the end of training as it would be when convergence to the true Q-values is achieved. Lastly, just as for the Gridworld 3x3 domain, none of the hidden layers of 4-layer DQNs learn internal representations that are more similar to the coarsest Markov state representation than the output-layer representation during training when the ground state is encoded via features (see Figure A.17).

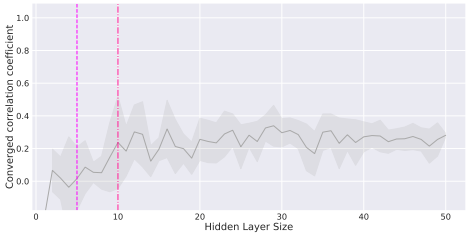


Figure A.16: Mean converged c_{Q^*} in the output layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the FrozenLake 4x4 (F) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

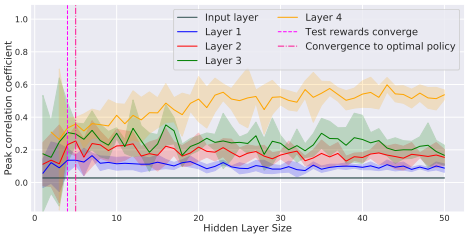


Figure A.17: Mean peak $c_{K(d_{fix})}$ in each layer with 95%-confidence intervals for each hidden layer size and constant input-layer $c_{K(d_{fix})}$ for a 4-layer DQN for the FrozenLake 4x4 (F) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

A.1.3. Correlation Coefficients When the Test Rewards Converge, the Optimal Policy is Discovered, and Convergence to the Optimal Policy is Achieved

Besides the internal state representations that are learned by DQNs, we are also interested in the specific internal state representations required for the test rewards to converge, the optimal policy to be discovered, and convergence to the optimal policy to be achieved. Note that with the convergence of the test rewards we mean that all test episodes end successfully for a training episode and all subsequent training episodes. Convergence of the test rewards is especially interesting for the FrozenLake domains, for which simply reaching the goal state is difficult due to the holes, stepping into which ends an episode.

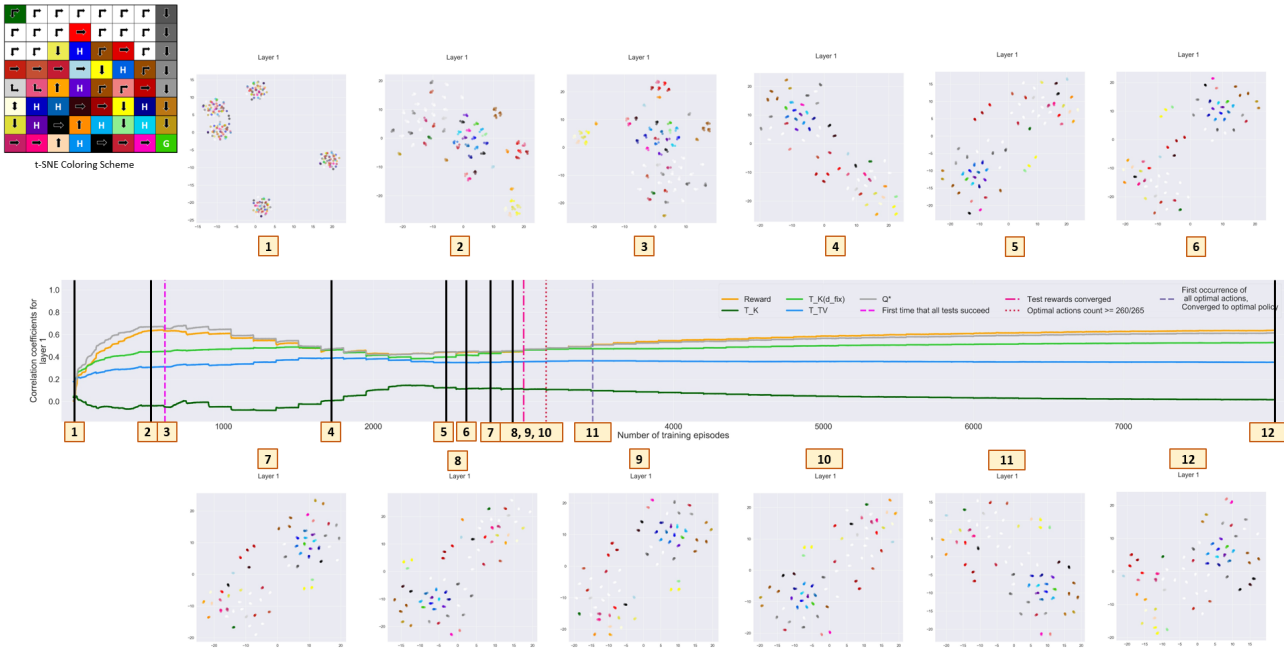
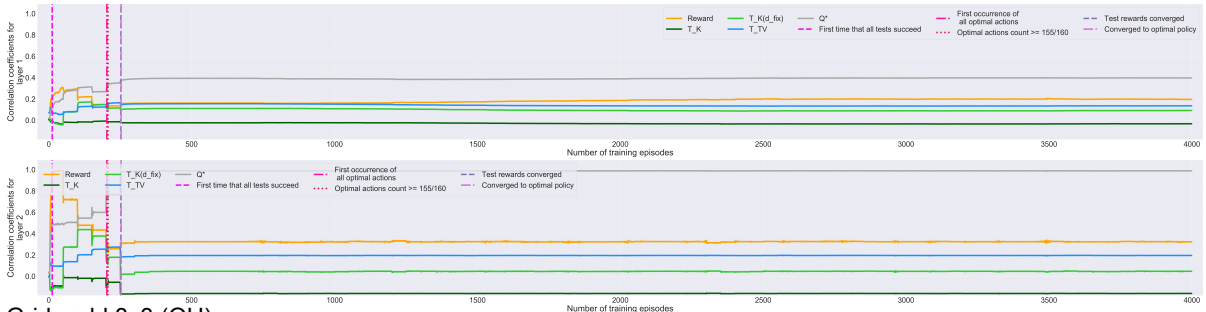


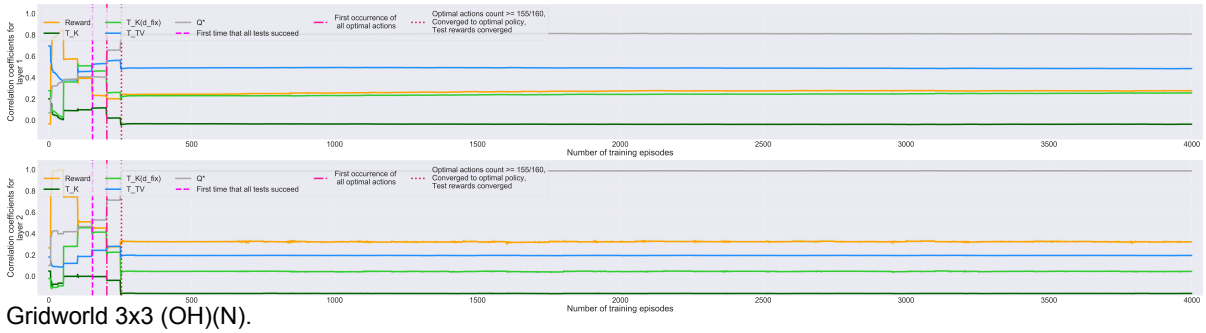
Figure A.18: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the first layer of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 8x8 (OH)(F-OH) domain.

Chapter 3 shows that the internal state representations learned in hidden layers depend on various factors such as the state encoding, state space size and the network capacity, and that the precise coarsest Markov state representation is not learned for any of our experiments. Therefore, looking at the values of correlation coefficients alone does not allow drawing any conclusions regarding whether all tests succeed, a DQN has discovered the optimal policy, or convergence to the optimal policy has been achieved. This becomes clearly visible in Figures A.19, A.20 and A.21, which depict the correlation coefficients during training for 2-layer DQNs with a hidden layer size of 50 for the Gridworld 3x3, FrozenLake 4x4, FrozenLake 8x8 and Gridworld 5x5 domains for all types of state encoding. Nevertheless, we can make a few noteworthy observations:

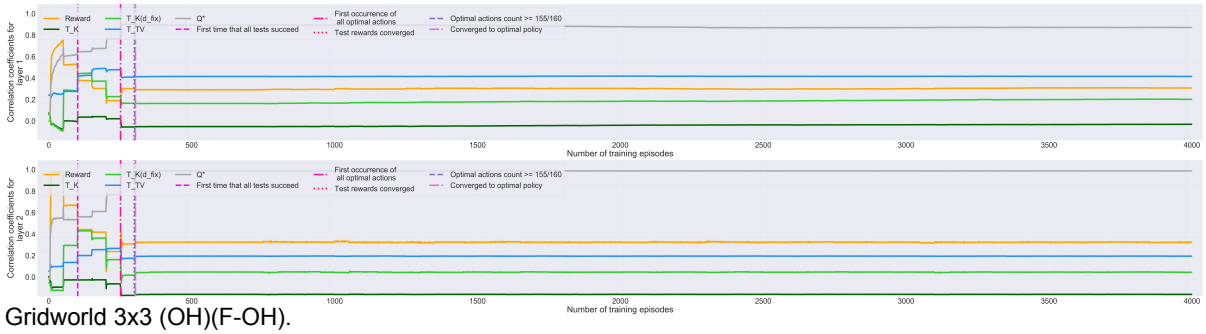
- **All correlation coefficients have largely converged when a DQN for a Gridworld domain converges to the optimal policy**. c_{TV}, c_K, c_{Rew}, $c_{K(d_{fix})}$ and c_{Q^*} have mostly converged in both layers when a 2-layer DQN converges to the optimal policy for a Gridworld domain, but not for a FrozenLake domain. This is the case, because a DQN for a Gridworld domain has to have learned rather accurate Q-values to be able to always act optimally, since states have very similar Q-values for optimal and non-optimal actions in the Gridworld domains. In the FrozenLake domains, however, the Q-values for optimal and non-optimal actions differ to a much larger extent. Therefore, a DQN for a FrozenLake domain may converge to the optimal policy even before the precise Q-values have been learned. This means that the internal state representations may afterwards still be modified to a larger degree than for the Gridworld domains. Notice that if the network capacity is large, such subsequent changes for DQNs for the FrozenLake domains are generally limited to the output layer, because the first layer then does not have to form close to a Q^*-irrelevance abstraction.



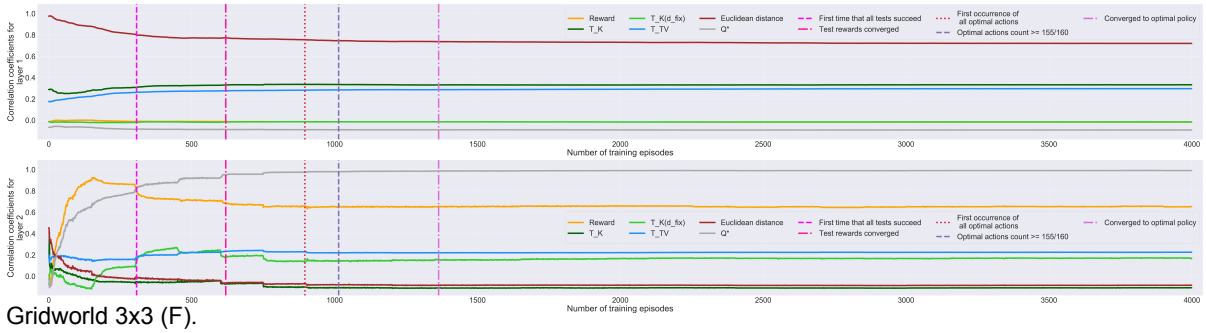
(a) Gridworld 3x3 (OH).



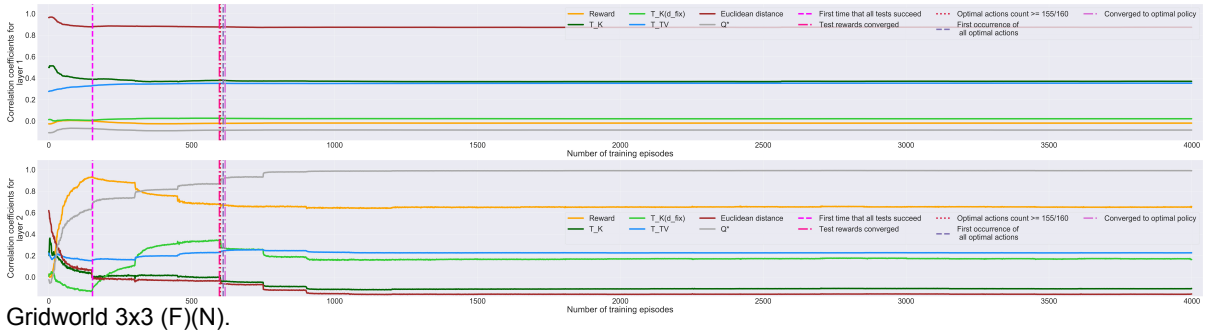
(b) Gridworld 3x3 (OH)(N).



(c) Gridworld 3x3 (OH)(F-OH).

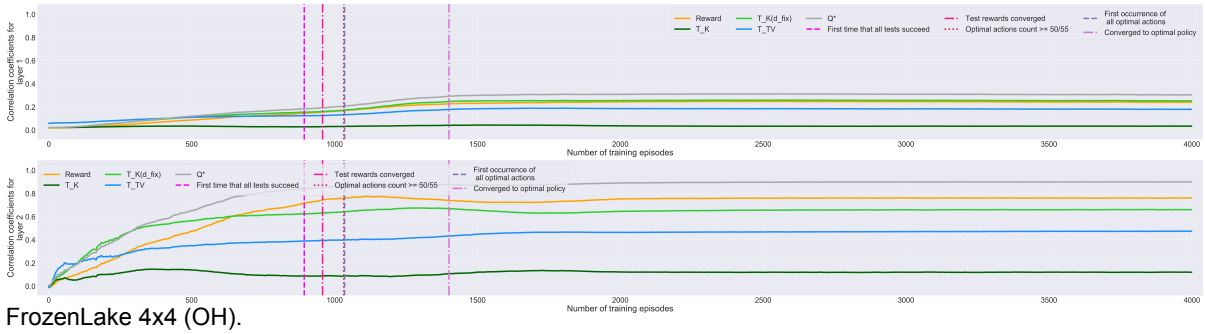


(d) Gridworld 3x3 (F).

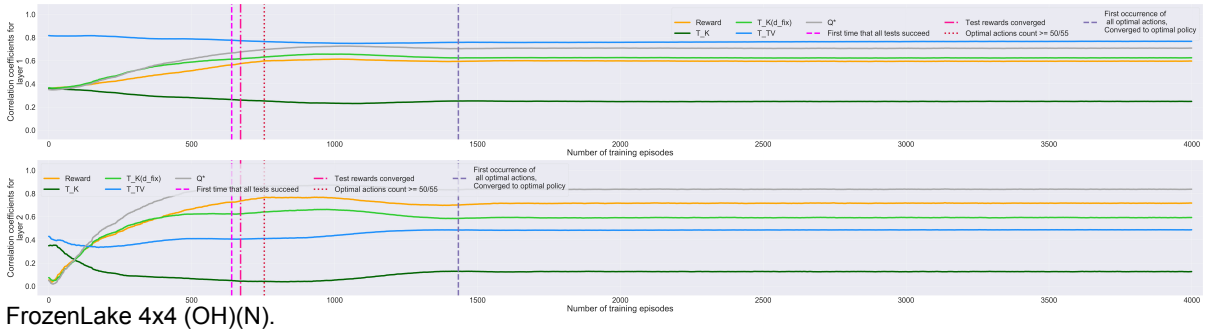


(e) Gridworld 3x3 (F)(N).

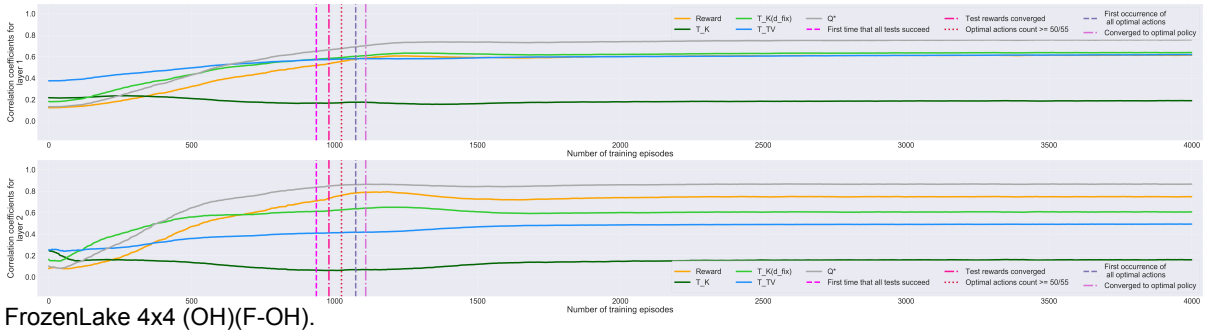
Figure A.19: Correlation coefficients for the layers of 2-layer DQNs for the Gridworld 3x3 domain with different forms of state encoding. The hidden layer size is equal to 50.



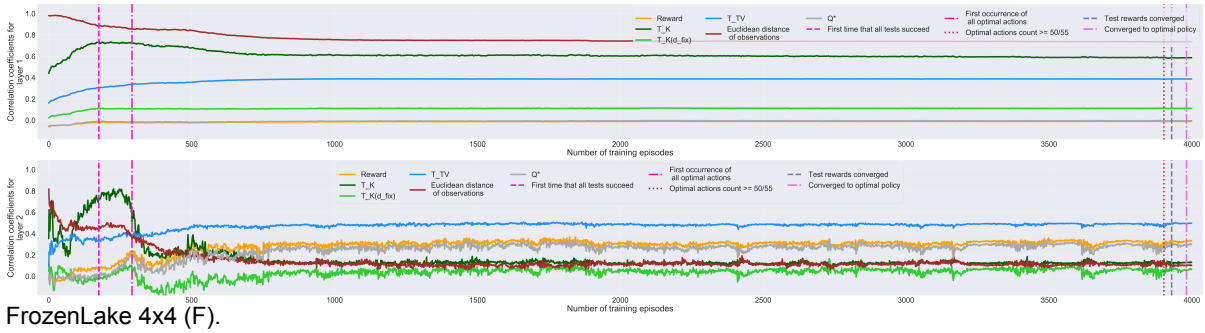
(a) FrozenLake 4x4 (OH).



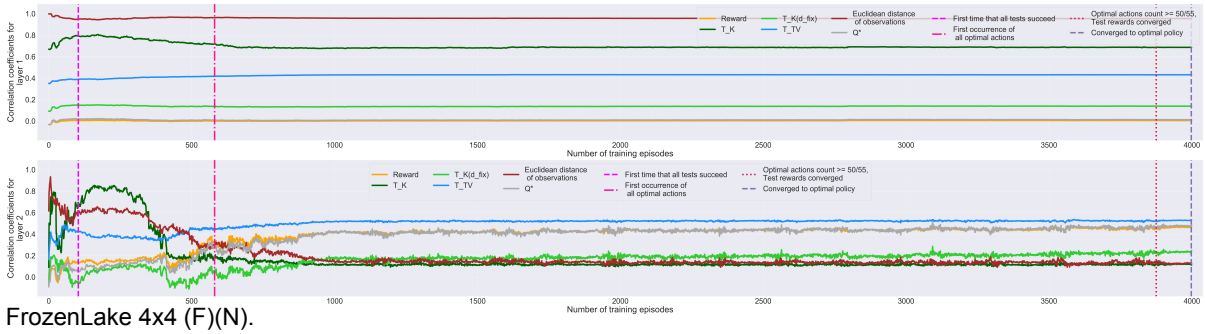
(b) FrozenLake 4x4 (OH)(N).



(c) FrozenLake 4x4 (OH)(F-OH).

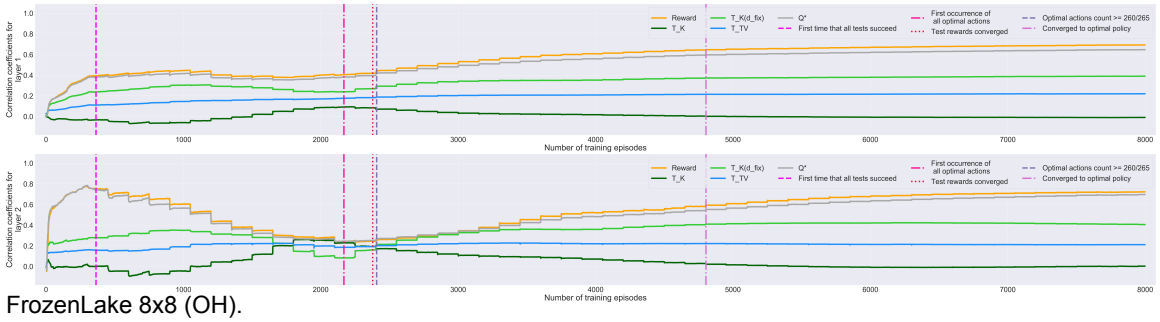


(d) FrozenLake 4x4 (F).

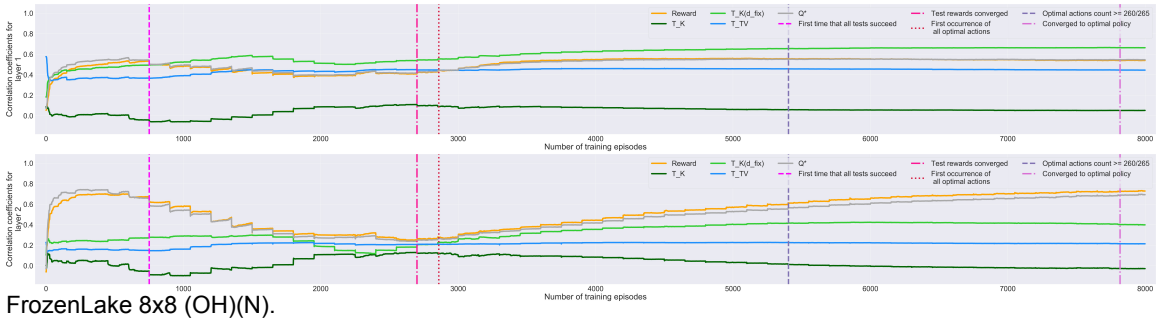


(e) FrozenLake 4x4 (F)(N).

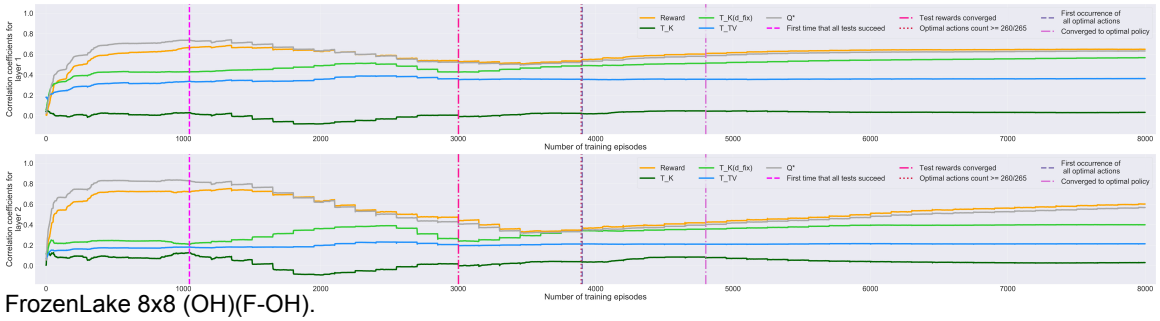
Figure A.20: Correlation coefficients for the layers of 2-layer DQNs for the FrozenLake 4x4 domain with different forms of state encoding. The hidden layer size is equal to 50.



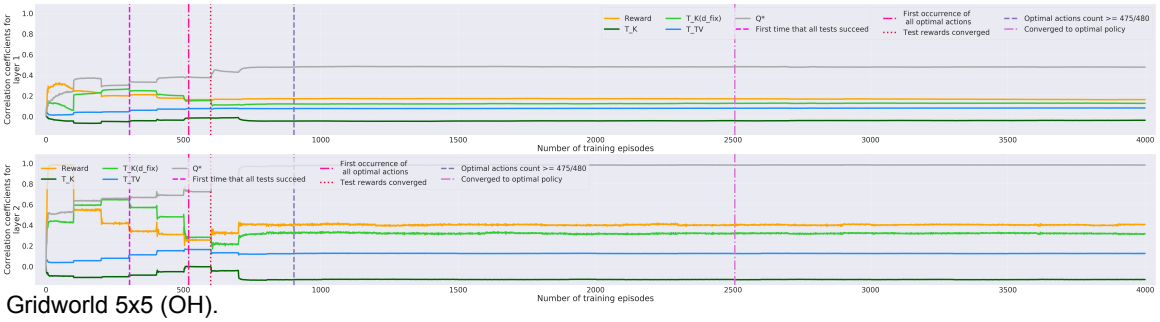
(a) FrozenLake 8x8 (OH).



(b) FrozenLake 8x8 (OH)(N).



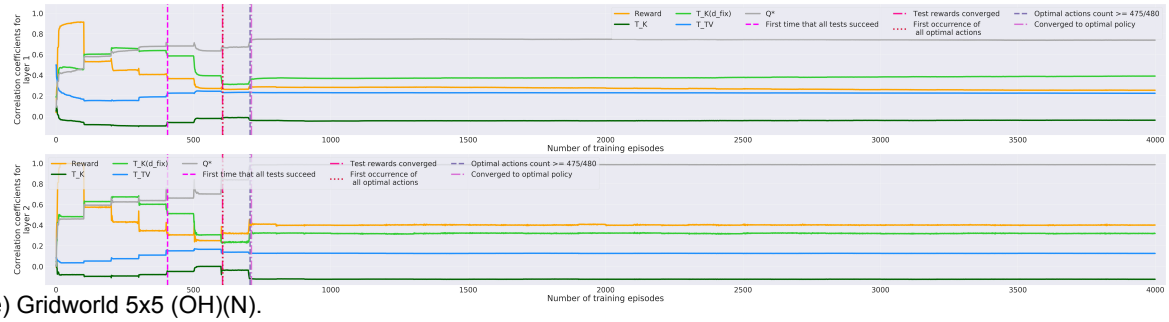
(c) FrozenLake 8x8 (OH)(F-OH).



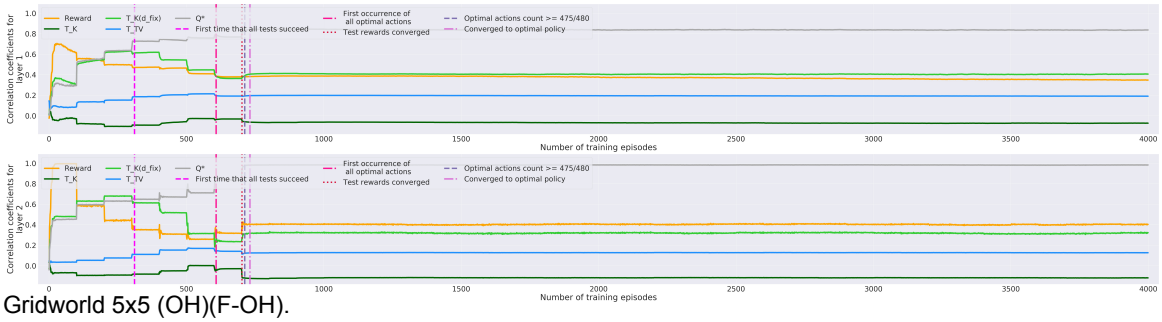
(d) Gridworld 5x5 (OH).

Figure A.21: Correlation coefficients for the layers of 2-layer DQNs for the FrozenLake 8x8 and Gridworld 5x5 domains with different forms of state encoding. The hidden layer size is equal to 50.

- **A DQN converges to the optimal policy soon after c_{TV} no longer changes significantly.** Figures A.19, A.20 and A.21 visualize that c_{TV} typically no longer changes significantly when a DQN converges to the optimal policy even for the FrozenLake domains. Recall that c_{TV} takes on a value of 1 for our domains when states are mapped to the same activations if and only if they belong to the same ground state and all inter-cluster distances are equal. Hence, c_{TV} has mostly converged when a DQN for the FrozenLake domain converges to the optimal policy, because states from the same ground states already are mapped to very similar activations and the inter-cluster distances already are very far from being equidistant at that point. For instance, Figure A.18 shows for the first layer of a 2-layer DQN for the FrozenLake 8x8 (OH)(F-OH) domain that states corresponding to the same ground state are mapped to very similar activations from t-SNE plot 5 onward and that only inter-cluster distances vary slightly in subsequent t-SNE plots as the network learns the precise Q-values.



(e) Gridworld 5x5 (OH)(N).



(f) Gridworld 5x5 (OH)(F-OH).

Figure A.21: Correlation coefficients for the layers of 2-layer DQNs for the FrozenLake 8x8 and Gridworld 5x5 domains with different forms of state encoding. The hidden layer size is equal to 50.

A.1.4. Using Bisimulation Metrics to Choose an Adequate Network Capacity

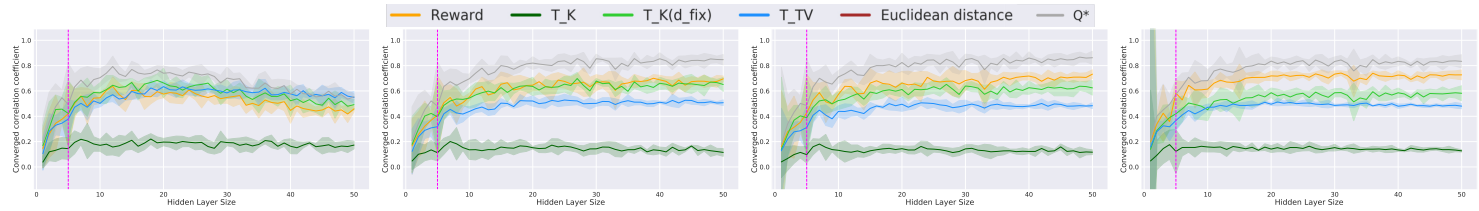
The analysis of the state representations that a DQN forms in each of its layers in Chapter 3 does already take different network capacities into consideration. This section, however, now explicitly explores whether bisimulation-based correlation coefficients during or at the end of training allow to draw conclusions regarding the adequacy of the capacity of a DQN, which is defined by the number of layers and number of nodes per layer. Specifically, it would be interesting if one could say whether a DQN with a certain hidden layer size and number of layers has insufficient, adequate, or too much capacity after computing these measures. An adequate network capacity is important, because a network with insufficient capacity may not be able to learn an optimal policy, and a network with too much capacity requires more memory and training and testing time than necessary and may fail to converge.

A.1.4.1 Hidden Layer Size

Ideally, the values of correlation coefficients alone would indicate whether the used hidden layer size is too small, adequate, or larger than necessary. Yet, as discussed in Chapter 3, the precise coarsest Markov state representation is not learned during any of our experiments. Furthermore, the values of the correlation coefficients for other types of internal state representations differ based on various aspects such as the state encoding and characteristics of the domain itself. This renders drawing conclusions regarding the adequacy of the hidden layer size based on solely the values of the correlation coefficients impossible. While we found the values of the correlation coefficients during as well as at the end of training to be contingent on the hidden layer size in Chapter 3, the patterns we discovered do not make it obsolete to train a DQN with various hidden layer sizes to find an adequate one. For instance, we saw that $c_{K(d_{fix})}$ tends to be highest at the end of training for slightly larger-than-necessary hidden layer sizes. However, to find the hidden layer size for which $c_{K(d_{fix})}$ is highest at the end of training, one has to train DQNs with different hidden layer sizes. Computing $c_{K(d_{fix})}$ thus does not offer any shortcut to choosing an adequate hidden layer size.

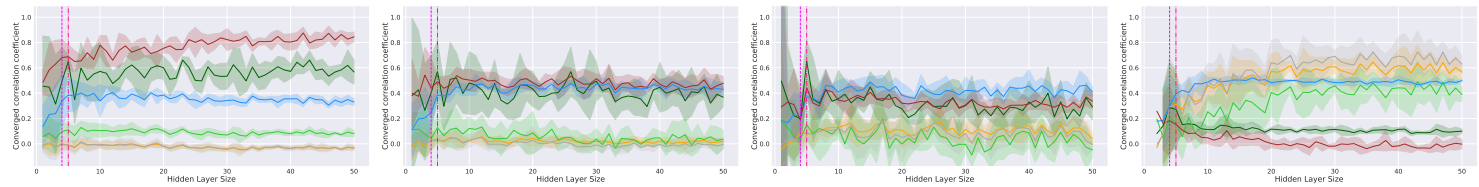
A.1.4.2 Number of Hidden Layers

Adequately many vs. more hidden layers than necessary. Just as in the context of choosing an adequate number of hidden nodes, the values of the correlation coefficients alone do not indicate whether or not the number of hidden layers is adequate. What we do see is that for just-right and slightly larger hidden layer sizes, all four layers of 4-layer DQNs typically form very similar internal state representations and thus have similar values for the correlation coefficients at the end of training when only a single hidden layer is needed, which is the case for our domains when the ground state is one-hot



First layer. Second layer. Third layer. Output layer.

(a) FrozenLake 4x4 (OH)(F-OH).



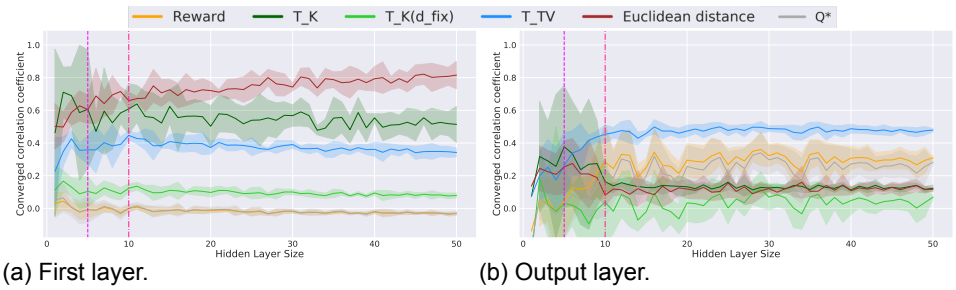
First layer. Second layer. Third layer. Output layer.

(b) FrozenLake 4x4 (F).

Figure A.22: Mean converged c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV}, c_E and c_{Q^*} in each layer with 95%-confidence intervals for each hidden layer size for 4-layer DQNs for the FrozenLake 4x4 (OH)(F-OH) and the FrozenLake 4x4 (F) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer size for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

encoded (see Figure A.22a). When multiple hidden layers are useful because they necessitate a lower total number of hidden nodes, as when the ground state is encoded via features for the FrozenLake 4x4 domain, the values of the correlation coefficients at the end of training in the layers differ to a larger extent for just-right and slightly larger hidden layer sizes (see Figure A.22b).

Yet, looking at whether or not the values of the correlation coefficients at the end of training are similar for the layers alone does not allow drawing conclusions regarding the adequacy of the number of layers. The reason is that the differences between the internal state representations formed in the hidden layers increase as the hidden layer size becomes larger than necessary, even when not all hidden layers are needed. Consequently, large differences between the internal state representations present in hidden layers at the end of training could mean either that the number of hidden layers is adequate, or that fewer and smaller hidden layers should be utilized. Even if one additionally finds that the learned representations in earlier hidden layers are very close to the state encoding, one can only conclude that smaller hidden layers should be used. This is the case, because it occurs both when the number of hidden layers is adequate and when there are more hidden layers than needed that the formed first-layer representation is increasingly similar to the state encoding for larger-than-necessary hidden layer sizes.



(a) First layer. (b) Output layer.

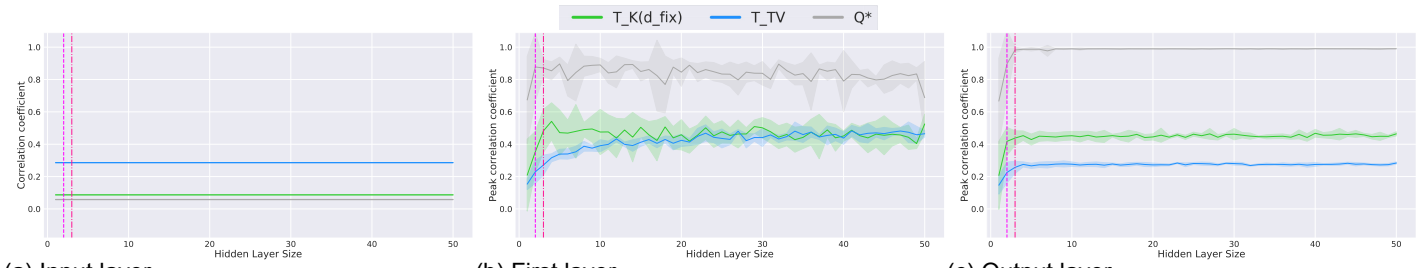
Figure A.23: Mean converged c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV}, c_E and c_{Q^*} in each layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the FrozenLake 4x4 (F) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

Adequately many vs. too few hidden layers. Our experimental results suggest that while using a single hidden layer for a DQN for the FrozenLake 4x4 domain when the ground state is encoded via features allows the DQN to converge to the optimal policy, even a hidden layer size of 50 is not yet sufficient for the DQN to converge to the true Q-values. This is mirrored in Figure A.23 by the fact

that c_{Q^*} in the second layer takes on a final value much lower than the one near 1 that is has for a Q^*-irrelevance abstraction. When employing three hidden layers instead, a hidden layer size of 10 already causes more accurate Q-values to be learned than for any tested hidden layer size for a 2-layer DQN. Thus, for a fixed total number of hidden nodes, a 4-layer DQN performs much better than a 2-layer DQN for the FrozenLake 4x4 domain when the ground state is encoded via features. Hence, one should use more than just a single hidden layer for this domain and type of ground state encoding.

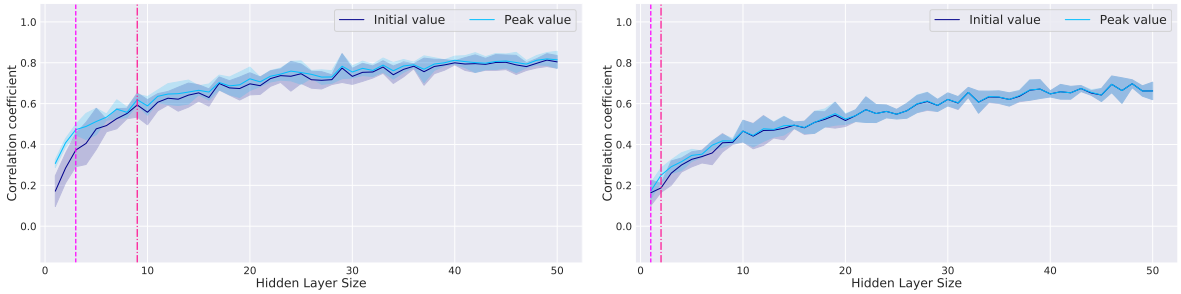
Yet, based on the values of the correlation coefficients at the end of or during training of a 2-layer DQN for the FrozenLake 4x4 domain, it does not become clear that utilizing more hidden layers would ameliorate the training process. In fact, the first-layer correlation coefficients at the end of training are very similar for both 2-layer DQNs and 4-layer DQNs. This becomes evident when comparing Figure A.22b to Figure A.23. Furthermore, while the final values of $c_{K(d_{fix})}$ and c_{Rew} are much lower for the output layer of a 2-layer DQN than for the output layer of a 4-layer DQN, they are also lower for too small hidden layers for a 4-layer DQN than for adequate hidden layer sizes for a 4-layer DQN. Therefore, the final output-layer values of $c_{K(d_{fix})}$ and c_{Rew} are not indicative of whether more hidden layers or more hidden nodes are most adequate. Lastly, notice that the precise values of those correlation coefficients for a Q^*-irrelevance abstraction are domain-dependent, which means that unless one knows their values for a Q^*-irrelevance abstraction for a domain beforehand, one cannot even compare the final output-layer values for $c_{K(d_{fix})}$ and c_{Rew} to the ideal ones.

A.1.5. Further Figures



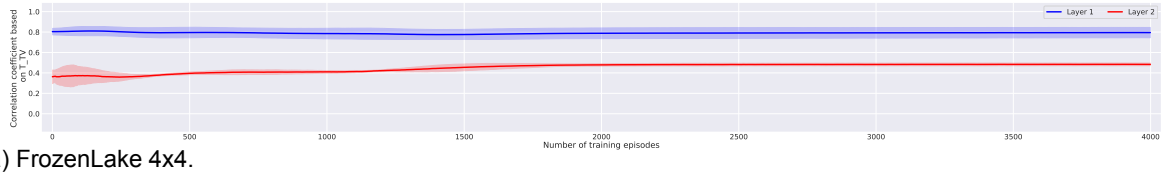
(a) Input layer. (b) First layer. (c) Output layer.

Figure A.24: Mean peak $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in each layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the Gridworld 3x3 (OH)(F-OH) domain. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.



(a) FrozenLake 4x4. (b) Gridworld 3x3.

Figure A.25: Mean initial and peak c_{TV} in the first layer with 95%-confidence intervals for each hidden layer size for 2-layer DQNs for the FrozenLake 4x4 and the Gridworld 3x3 domain for the (OH)(N) state encoding. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.



(a) FrozenLake 4x4.

(b) Gridworld 3x3.

Figure A.26: Mean c_{TV} in the layers of 2-layer DQNs for the FrozenLake 4x4 and Gridworld 3x3 domains for the (OH)(N) state encoding. Values are based on 5 repetitions and 95%-confidence intervals are shown. The hidden layer size is equal to 50.

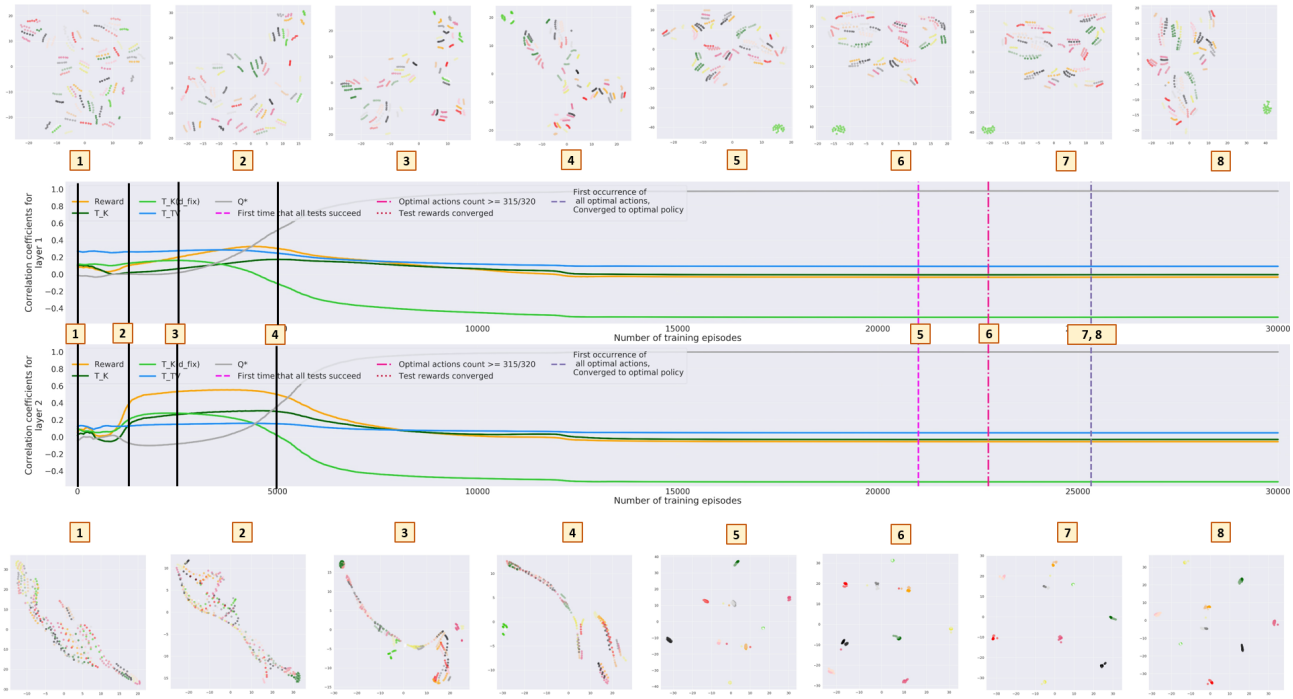
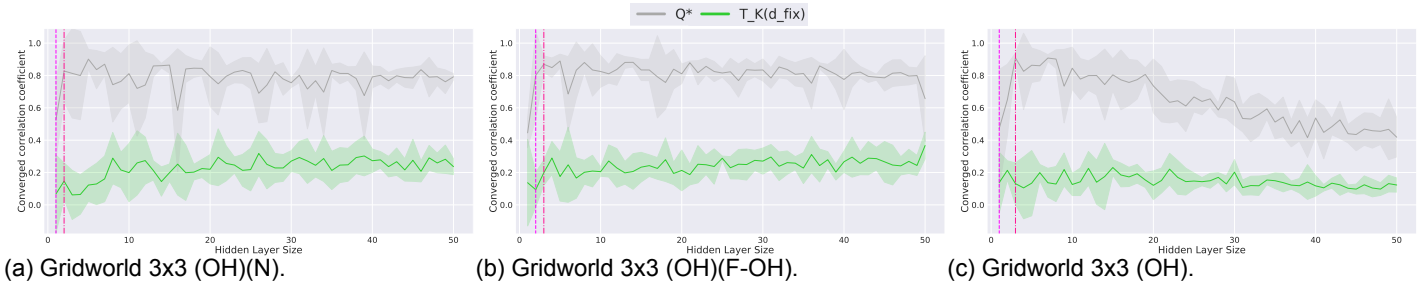


Figure A.27: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN for the Gridworld 3x3 (Aug) (OH)(N) domain. The hidden layer size is equal to 6.



(a) Gridworld 3x3 (OH)(N). (b) Gridworld 3x3 (OH)(F-OH). (c) Gridworld 3x3 (OH).

Figure A.28: Mean converged $c_{K(d_{fix})}$ and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the Gridworld 3x3 domain with different forms of state encoding. Values are based on 5 repetitions. The first and second vertical line indicate the smallest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.

A.2. Impact of Markovianity on Learning Speed and Consistency

Section A.2.1 discusses how we set d_B^{max} for auxiliary losses based on d_{fix} and d'_{fix} for our experiments, and Sections A.2.2, A.2.3 and A.2.4 contain further results for the experiments with auxiliary losses based on d_{fix}, d'_{fix} and T_{TV}, respectively.

A.2.1. Setting d_B^{max}

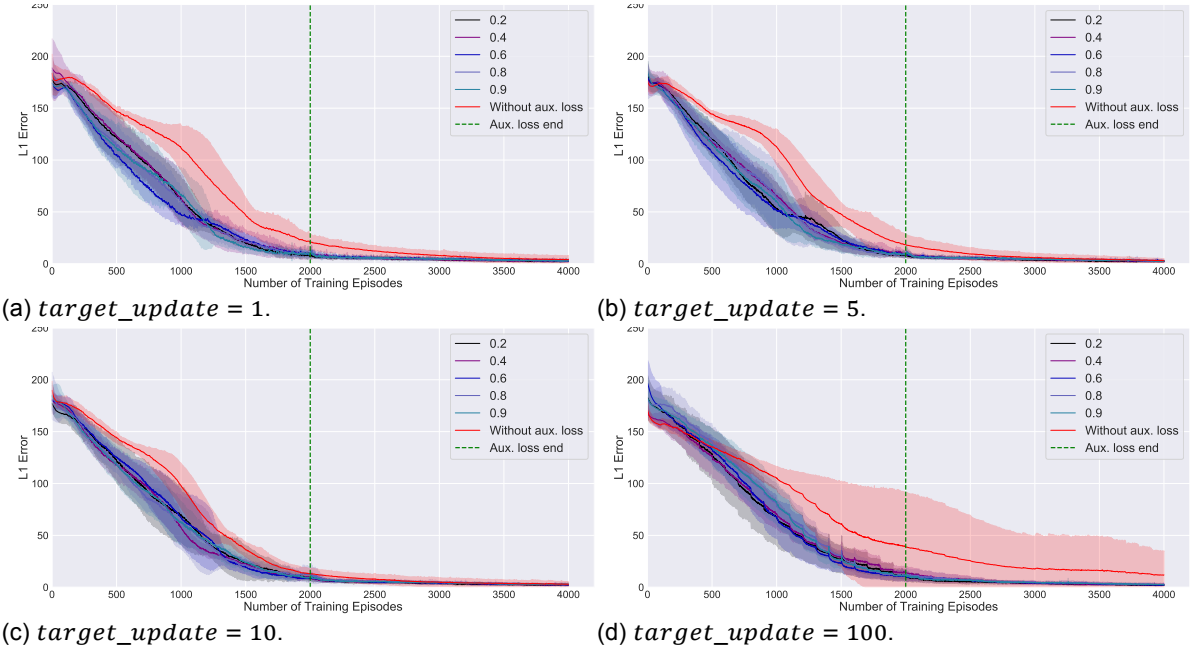
While d_B^{max} is clearly equal to 1 when T_{TV} is utilized, we have two options for reasonably setting d_B^{max} when d_{fix} or d'_{fix} are employed for the auxiliary loss. First, we could set d_B^{max} to the maximum appearing distance based on d_{fix} or d'_{fix} for a domain. However, this makes it difficult to compare the impact of varying values for d_E^{max} for different domains, because states with similar bisimulation-based distances are not necessarily pushed to roughly equally distant activations for different domains. Thus, we set d_B^{max} equal to 1 for both Gridworld 3x3 and FrozenLake 4x4 when using an auxiliary loss based on d_{fix} or d'_{fix}. Yet, when comparing the use of d'_{fix} to the one of d_{fix}, it is important to take the large discrepancy between the maximum arising distances into consideration. For instance, given that the largest occurring value for d'_{fix} for FrozenLake 4x4 is 24.5 times as high as the one for d_{fix}, d_E^{max} should be set 24.5 times higher for d_{fix} than for d'_{fix} to obtain comparable results.

A.2.2. Auxiliary Loss Based on d_{fix}

Section A.2.2.1 and Section A.2.2.2 provide further experimental results for the FrozenLake 4x4 (OH) and the Gridworld 3x3 (OH) domain, respectively.

A.2.2.1 FrozenLake 4x4

In the following, we analyze the impacts of the target network update frequency and the hyperparameter settings for the auxiliary loss using the FrozenLake 4x4 domain as example.



(a) $target\_update = 1$. (b) $target\_update = 5$.

(c) $target\_update = 10$. (d) $target\_update = 100$.

Figure A.29: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss, varying target network update frequencies, and terminating the usage of the auxiliary loss after training episode $2,000$, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{256 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

Target network update frequency. While introducing the auxiliary loss improves the L_1-error during training and allows for earlier convergence for all tested target network update frequencies, the magnitudes of improvement at various stages of training vary across different values for this hyperparameter:

- The auxiliary loss leads to less improvement in the L_1-error or even slight deterioration at the very beginning of training while the target network has not yet been updated $n-1$ times, where n is the maximum number of transitions Q-values are based on. Hence, the improvement in L_1-error is largest at the beginning of training for small values for the $target\_update$-parameter (see Figure A.29). This is the case, because the auxiliary loss prevents precise grouping based on rewards considering few transitions. Yet, since the auxiliary loss does not impact the output-layer representation, it does not yet allow states to be fully clustered based on Q-values either. As learning to predict the immediate rewards already leads to rather accurate Q-values due to the high proportion of terminal states in the FrozenLake 4x4 domain, this therefore causes the L_1-error to be slightly higher at the beginning of training when the auxiliary loss is added than when it is not used[7].

- The auxiliary loss greatly reduces the variance and the magnitude of the L_1-error towards the end of training especially for values much higher or lower than the best tested value of 10 for the $target\_update$-parameter (see Figure A.29). The result is that the different settings for the target network update frequency now perform similarly well. This shows that introducing the auxiliary loss can render the training process more stable and more robust to different settings for hyperparameters such as the target network update frequency.

Hyperparameters of the auxiliary loss. To obtain the best possible improvement and no deterioration of the L_1-error during training, it is important to choose appropriate values for d_E^{max}, the duration, and the weight for the auxiliary loss:

- d_E^{max}. We from the start chose to render d_E^{max} dependent on the hidden layer size, because a smaller hidden layer implies that a DQN is less flexible and that very large target Euclidean distances may thus prevent the network from learning the true Q-values. More precisely, we set $d_E^{max} = \sqrt{w \times hidden\_layer\_size}$ and solely change the factor w. Our experimental results confirm that lower values for d_E^{max} are required for DQNs with smaller hidden layer sizes. Specifically, lower values for w are needed for such DQNs even though d_E^{max} already is lower for smaller DQNs for a fixed value for w. For instance, $w = 256$ leads to good results for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain as shown in Figure A.34d. However, values for w of 128 or 256 cause almost no improvement or even deterioration of the training process for a 2-layer DQN with a hidden layer size of 20 (see Figures A.34c and A.34d). Instead, $w = 32$ yields the largest improvements for a 2-layer DQN with a hidden layer size of 20 as visualized in Figure A.34b.

  When tuning d_E^{max} based on the hidden layer size, it is also important not to select too low of a value. For example, setting $w = 1$ for a 2-layer DQN with a hidden layer size of 50 deteriorates the L_1-error throughout training and induces later convergence to the true Q-values (see Figure A.36). Yet, besides too low and too large values for d_E^{max} leading to less improvement or even deterioration, multiple different settings for d_E^{max} enable good results. For instance, $w = 8$, $w = 32$, $w = 128$ and $w = 256$ all cause the auxiliary loss to significantly improve upon the L_1-error during the beginning of training for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain (see Figure A.34).

- *Duration*. In our experiments, we use a fixed decay rate of 0.9999 for the auxiliary loss and rather than varying the decay rate, we alter the number of training episodes during which the auxiliary loss is applied. The primary reason for this is that the naive computation of the auxiliary loss is expensive. Yet, it is also interesting to see what happens to the formed internal state representations when the auxiliary loss is no longer applied at varying times during training. The downside of this approach, however, is that if the auxiliary loss is still used with a large weight when its application is terminated, the learning targets may change abruptly.

  We find that the auxiliary loss should be terminated sooner for DQNs with smaller hidden layers. For instance, Figure A.34d visualizes that applying the auxiliary loss for $2,000$ training episodes

[7] See also Figure A.35, which shows that if the auxiliary loss is added to the training, states are less grouped based on Q-values in the first and the output layer at the beginning of training when the $target\_update$-parameter is set to 100 than when it is set to 1.

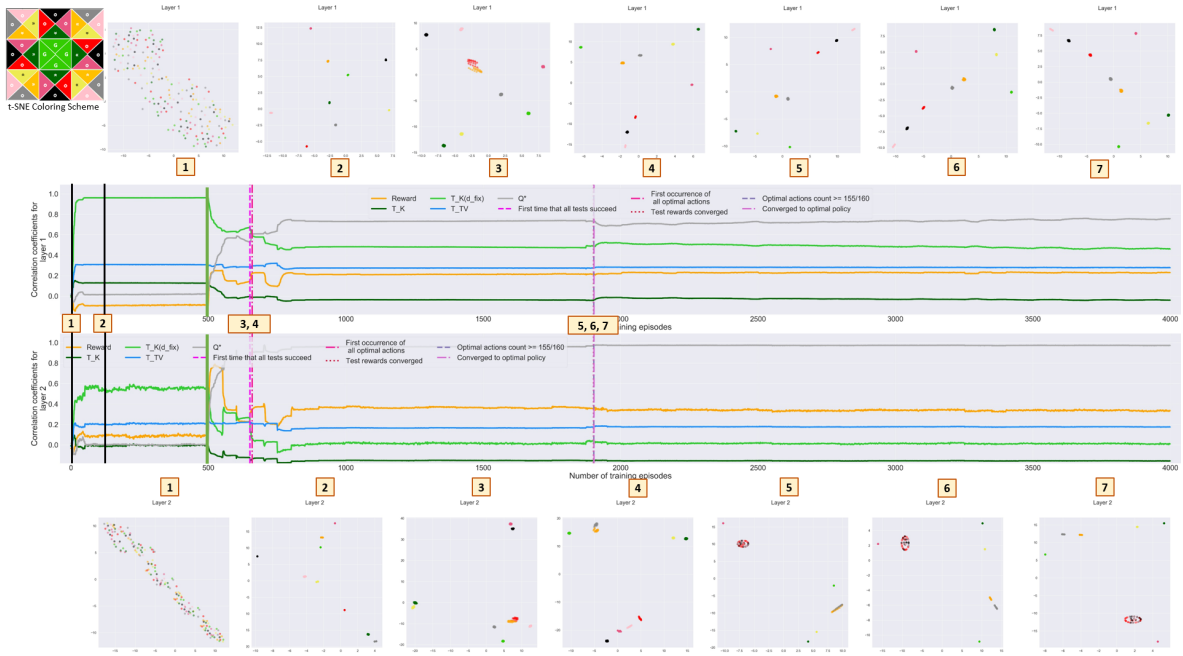


Figure A.30: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 3 for the Gridworld 3x3 (OH) domain. d_E^{max} is set to $\sqrt{32 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The auxiliary loss is applied with a weight of 0.1 and a decay rate of 0.9999 during the first 500 training episodes, the latter of which is indicated by the green vertical line.

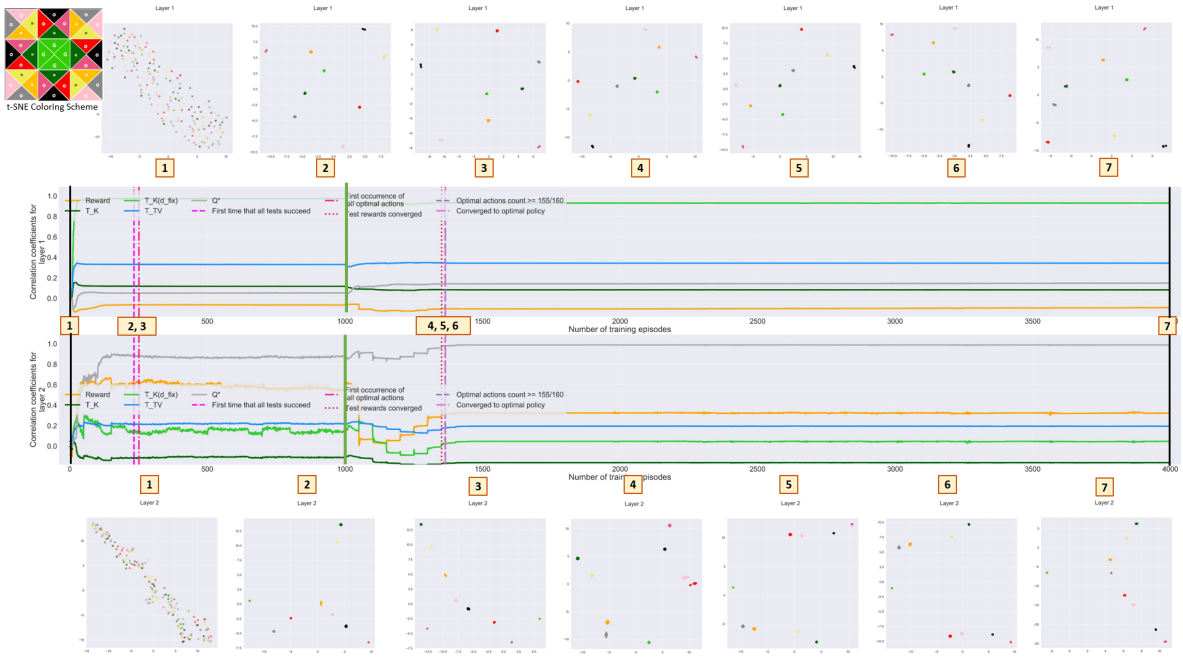
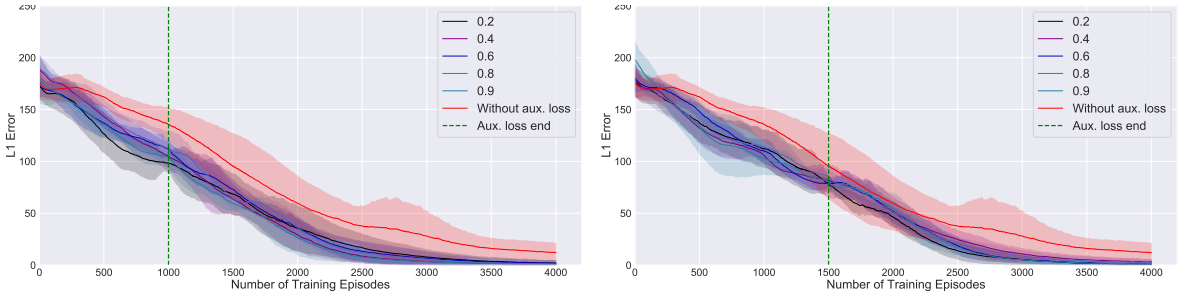


Figure A.31: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 5 for the Gridworld 3x3 (OH) domain. d_E^{max} is set to $\sqrt{128 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first 1,000 training episodes, the latter of which is indicated by the green vertical line.

yields good results for DQNs with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain. Yet, the same duration causes the L_1-error to ultimately deteriorate after initially improving for DQNs with a hidden layer size of 20, both for the same value of 256 for w and for the optimal value of 32 (see Figures A.34d and A.34b). Instead, terminating the auxiliary loss after 1,000 training episodes is useful for a DQN with a hidden layer size of 20 (see Figure A.32). This observation can be explained by the fact that DQNs with smaller hidden layers need to create an

abstraction coarser than the coarsest Markov state representation in their first layers to be able to converge to the true Q-values. Figure A.30 shows, for example, that if an auxiliary loss based on d_{fix} is applied to a 2-layer DQN for the Gridworld 3x3 (OH) domain with a hidden layer size of 3, the second layer cannot form a Q^*-irrelevance abstraction before the auxiliary loss is terminated. This is indicated by the very low value for c_{Q^*} while the auxiliary loss is used. The output layer of a 2-layer DQN with a hidden layer size of 5, however, can already create much closer to a Q^*-irrelevance abstraction while the auxiliary loss is applied (see Figure A.31). Nevertheless, even if the auxiliary loss is applied so long that the L_1-error increases at some point after initially decreasing, the L_1-error decreases again as soon as the auxiliary loss is terminated. This is, for instance, depicted in Figure A.34b when the auxiliary loss is utilized for $2,000$ training episodes.



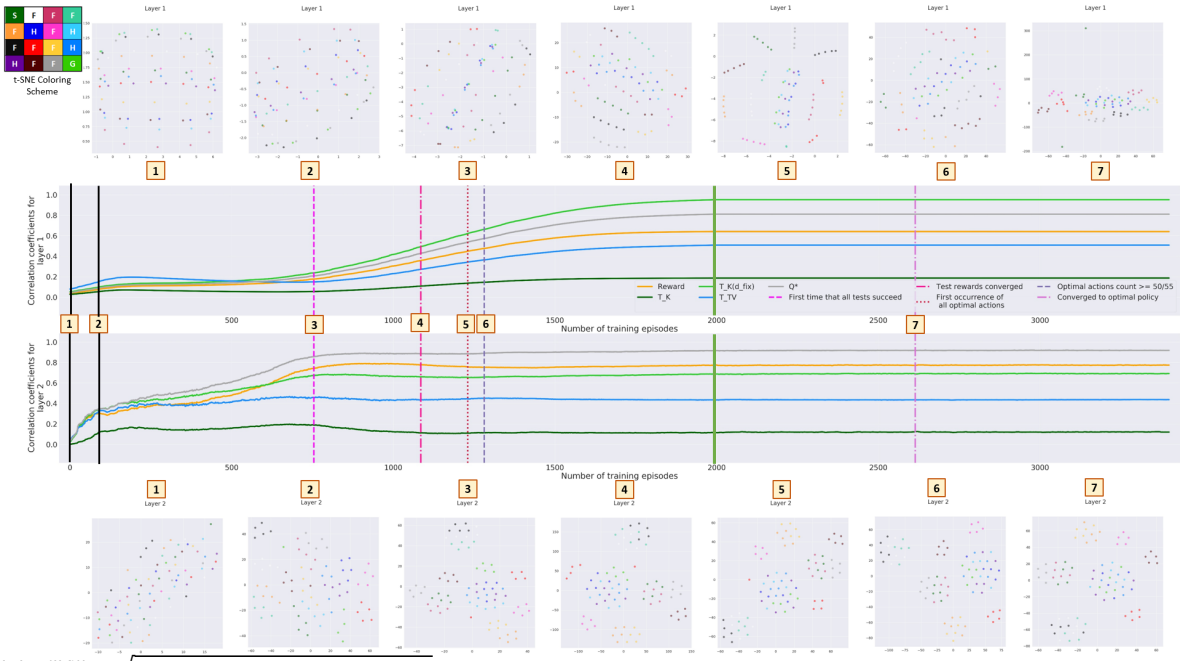
(a) Application for $1,000$ training episodes. (b) Application for $1,500$ training episodes.

Figure A.32: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 20 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss and $target\_update = 5$, and terminating the usage of the auxiliary loss after varying numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{32 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

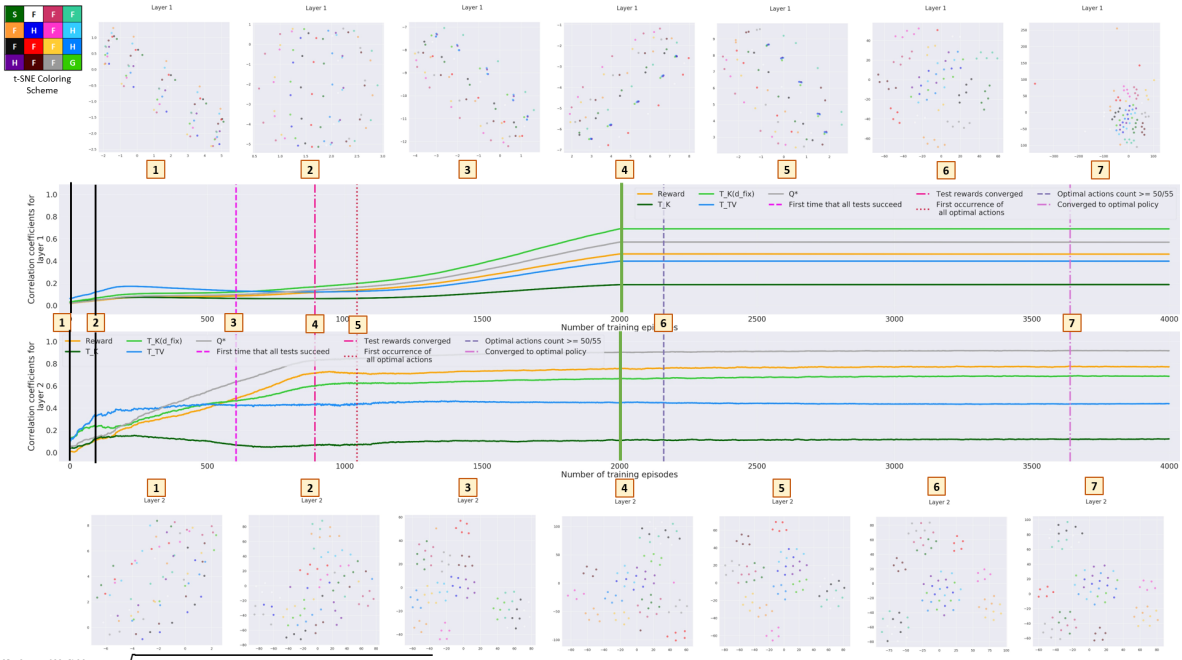
The duration should be adapted based on not only the hidden layer size, but also with respect to d_E^{max}. The reason is that more time is needed to group states differing solely in the superfluous feature value together in the first layer when d_E^{max} is higher. Note that the higher d_E^{max}, the more priority is given to adjusting the Euclidean distances between the activations of states with a large distance with respect to d_{fix} relative to tuning the Euclidean distances between the activations of states that are bisimilar. This is visualized in Figure A.33 for 2-layer DQNs for the FrozenLake 4x4 (OH) domain. For such a DQN, $c_{K(d_{fix})}$ is close to 1 in the first layer after $2,000$ training episodes if $w = 256$, but takes on a value below 0.8 at the same point during training if $w = 512$. Thereby, states differing only in the superfluous feature are mapped to much more similar activations in t-SNE plot 4 in Figure A.33a than in t-SNE plot 5 in Figure A.33b, both of which show the activations states are mapped to after about $1,100$ training episodes for the two different settings for d_E^{max}[8].

- *Weight.* We experimented with different weights of 0.2, 0.4, 0.6, 0.8 and 0.9 for the auxiliary loss. Our results show that for appropriate settings for the duration and d_E^{max}, the different weights lead to very similar outcomes. Yet, if the auxiliary loss is applied for too long or if too high or too low of a value for d_E^{max} is chosen, lower weights tend to cause less deterioration or even still improvement than larger weights. For instance, Figure A.34a visualizes that if the auxiliary loss is applied with $d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ to a 2-layer DQN for the FrozenLake 4x4 (OH) domain during the first $2,000$ training episodes, a weight of 0.2 still leads to lower L_1-errors during training than if no auxiliary loss is used. Higher weights, however, cause the L_1-error to increase at some point after initially dropping. In addition, very low weights such as 0.001 are necessary for DQNs with very small hidden layer sizes such as 3 to prevent exploding gradients.

[8] Bisimilar states with the same superfluous feature value are still grouped together early during training for high values for d_E^{max}. The reason is that such states are already mapped to relatively similar activations at the beginning of training for the (OH) state encoding.

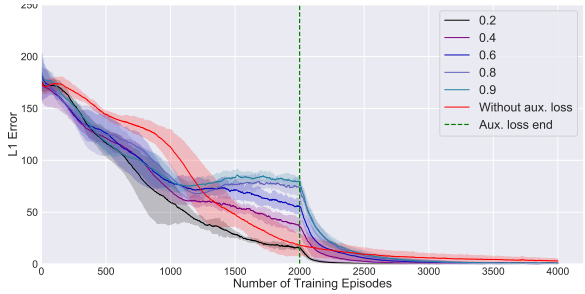


(a) $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$.



(b) $d_E^{max} = \sqrt{512 \times hidden\_layer\_size}$.

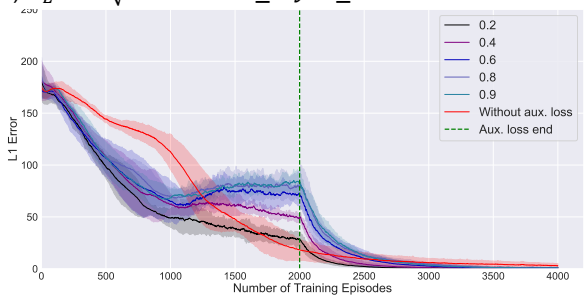
Figure A.33: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain for different values for d_E^{max}. d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first $2,000$ training episodes, the latter of which is indicated by the green vertical line.



$hidden\_layer\_size = 50.$

$hidden\_layer\_size = 20.$

(a) $d_E^{max} = \sqrt{8 \times hidden\_layer\_size}.$

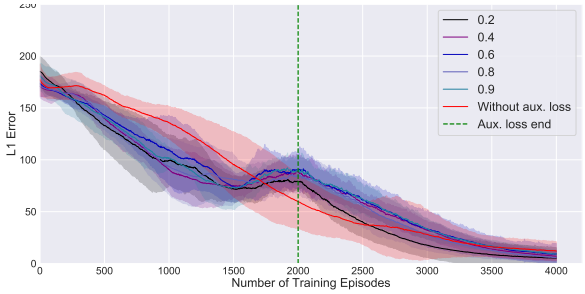


$hidden\_layer\_size = 50.$

$hidden\_layer\_size = 20.$

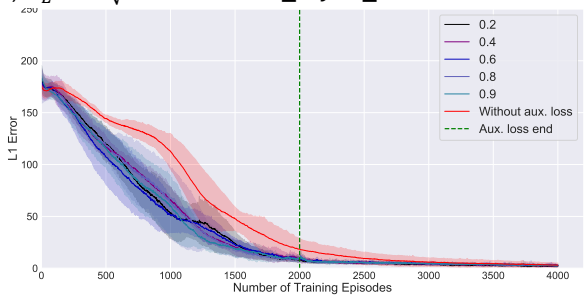
(b) $d_E^{max} = \sqrt{32 \times hidden\_layer\_size}.$

$hidden\_layer\_size = 50.$



$hidden\_layer\_size = 20.$

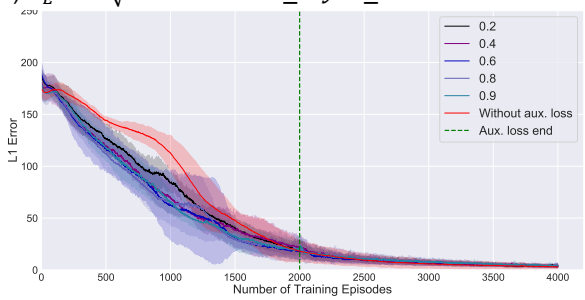
(c) $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}.$



$hidden\_layer\_size = 50.$

$hidden\_layer\_size = 20.$

(d) $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}.$



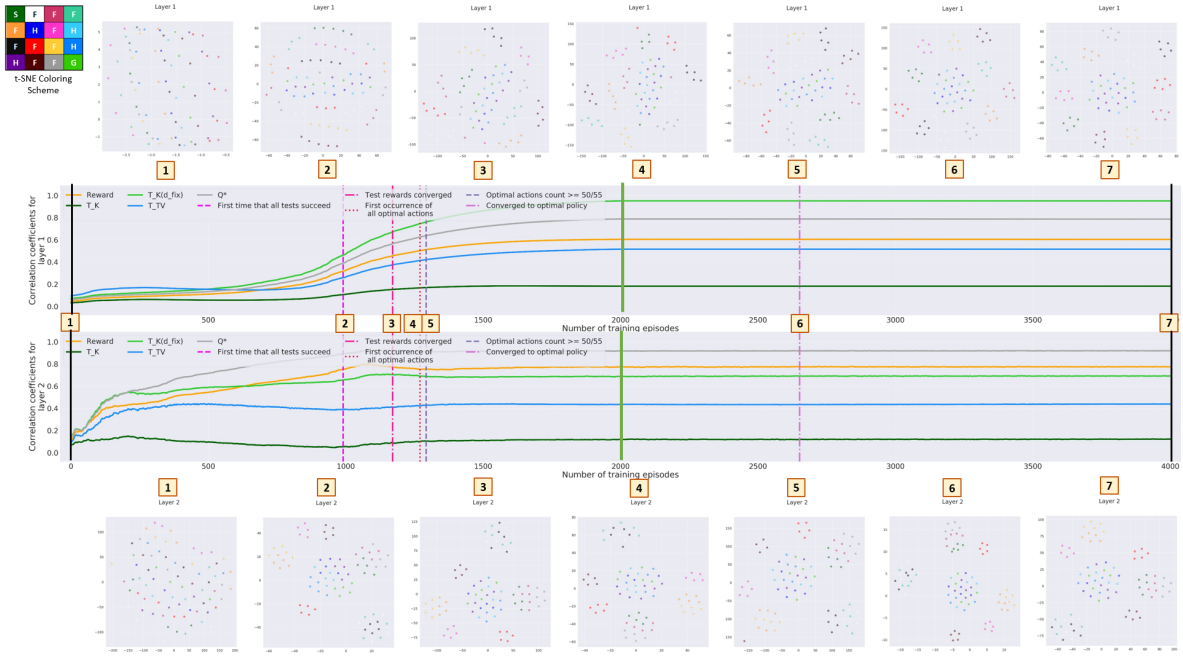
$hidden\_layer\_size = 50.$

$hidden\_layer\_size = 20.$

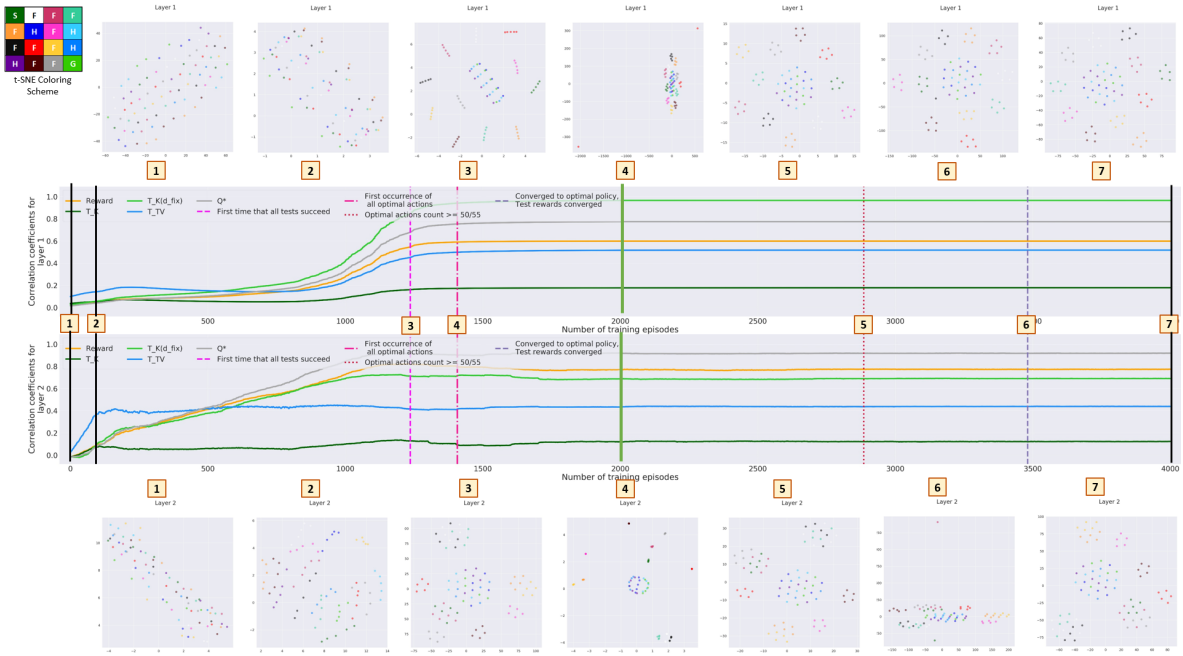
(e) $d_E^{max} = \sqrt{512 \times hidden\_layer\_size}.$

Figure A.34: L_1-error with respect to the true Q-values for a 2-layer DQN for the FrozenLake 4x4 (OH) domain when employing different hidden layer sizes, varying settings for d_E^{max}, different weights for the auxiliary loss and $target\_update = 5$, and terminating the usage of the auxiliary loss after training episode $2,000$, the latter of which is indicated by the green vertical line. d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

Further figures. Below, we provide further figures for experiments based on the FrozenLake 4x4 domain.

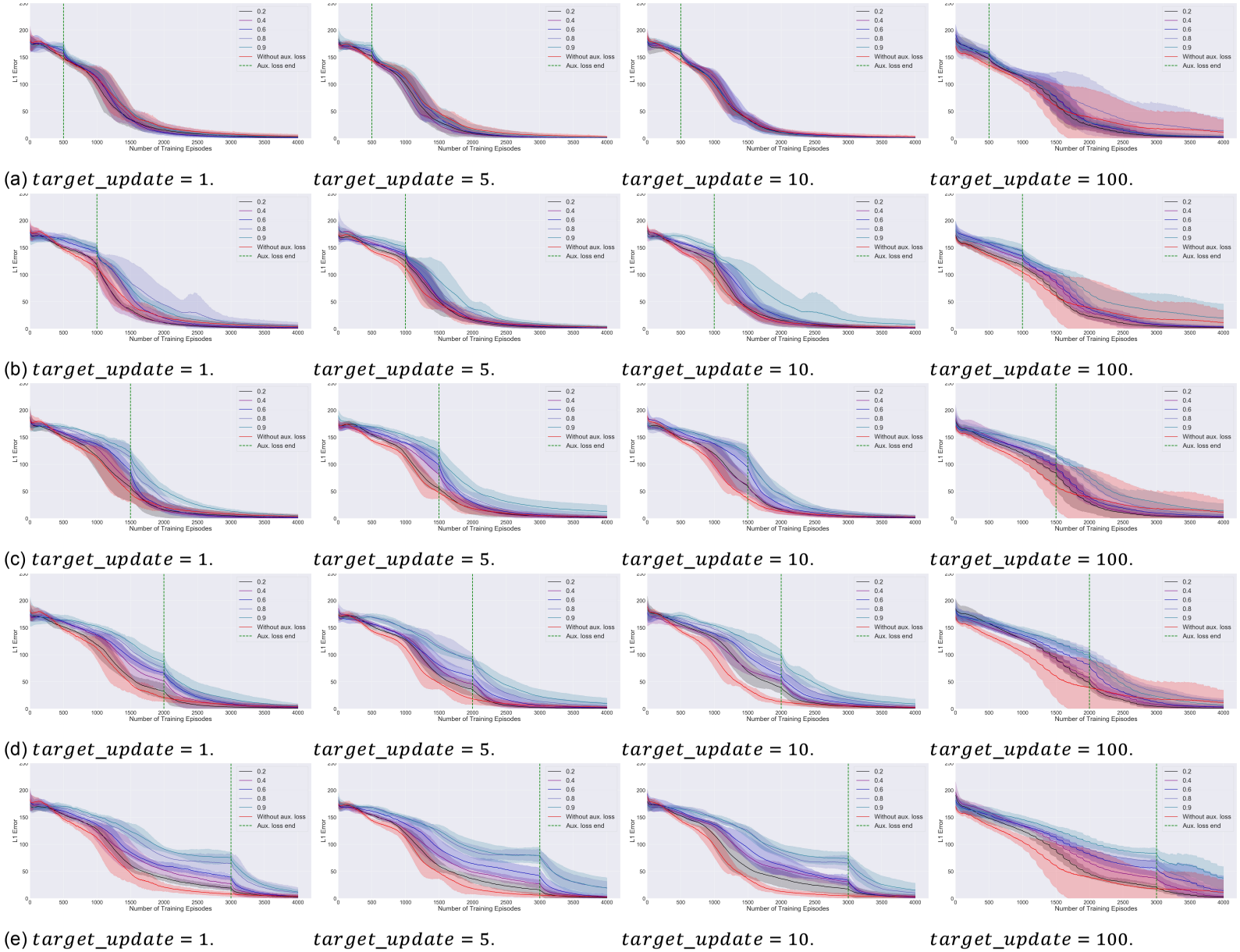


(a) $target\_update$ = 1.



(b) $target\_update$ = 100.

Figure A.35: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain for different target network update frequencies. $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first 2,000 training episodes, the latter of which is indicated by the green vertical line.



(a) $target\_update = 1.$ $target\_update = 5.$ $target\_update = 10.$ $target\_update = 100.$

(b) $target\_update = 1.$ $target\_update = 5.$ $target\_update = 10.$ $target\_update = 100.$

(c) $target\_update = 1.$ $target\_update = 5.$ $target\_update = 10.$ $target\_update = 100.$

(d) $target\_update = 1.$ $target\_update = 5.$ $target\_update = 10.$ $target\_update = 100.$

(e) $target\_update = 1.$ $target\_update = 5.$ $target\_update = 10.$ $target\_update = 100.$

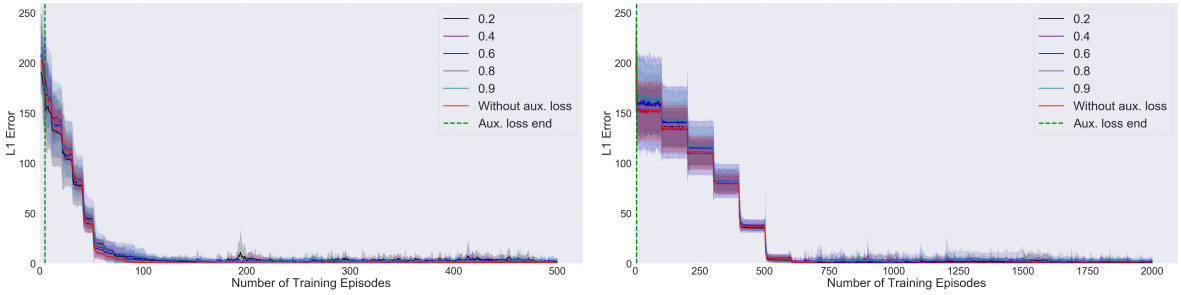
Figure A.36: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after varying numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

A.2.2.2 Gridworld 3x3

We found an auxiliary loss based on d_{fix} to generally improve upon the L_1-error during training when appropriate hyperparameters are chosen for the FrozenLake 4x4 (OH) domain in Chapter 4. Experiments on the Gridworld 3x3 (OH) domain, however, hardly show any improvement when applying this auxiliary loss. Figures A.37, A.39, A.40, A.41 and A.42 visualize for different values for d_E^{max} that introducing an auxiliary loss based on d_{fix} allows for no earlier convergence of a 2-layer DQN with a hidden layer size of 50, and only sometimes leads to slightly lower L_1-errors at the beginning of training.

The likely reason for this different outcome for the Gridworld 3x3 (OH) domain is that even without auxiliary loss, a 2-layer DQN converges to the true Q-values almost directly after the target network has been updated 5 times. Note that Q-values for the Gridworld 3x3 domain are based on at most 6 transitions. Hence, a DQN trained without auxiliary loss already converges as soon as is possible due to the fact that the target network provides the estimates of the Q-values of next states during training[9]. Adding an auxiliary loss that does not directly impact the output-layer representation therefore cannot lead to earlier convergence. The slight improvements in L_1-error that are observed for the Gridworld 3x3 (OH) domain when the auxiliary loss is introduced then arise, because adding the auxiliary loss allows mapping bisimilar states and hence states with the same Q-values closer together in both layers at the beginning of training than without auxiliary loss as depicted in Figure A.38 for a 2-layer DQN with a hidden layer size of 10.

Another interesting observation when applying this auxiliary loss to DQNs for the Gridworld 3x3 (OH) domain is that if the auxiliary loss is terminated when the network has almost or already converged to the true Q-values, the L_1-error sharply increases before subsequently decreasing again (see Figure A.37d). Thereby, the sudden increase in L_1-error is larger for higher values for d_E^{max} and for higher weights for the auxiliary loss. This phenomenon arises because of the downside of simply terminating the auxiliary loss rather than decaying it until it becomes negligible, as mentioned in the context of the impact of the hyperparameters of the auxiliary loss on DQNs trained for the FrozenLake 4x4 (OH) domain. Since the auxiliary loss still has a considerable impact on the overall loss when it is ended due to the decay rate of 0.9999 that we utilize, the gradients for the first-layer weights abruptly change when the auxiliary loss is stopped. The magnitude of the change thereby depends on the value for d_E^{max} as well as the weight of the auxiliary loss. Afterwards, the second-layer weights need to be adjusted to the suddenly altered first-layer representation.

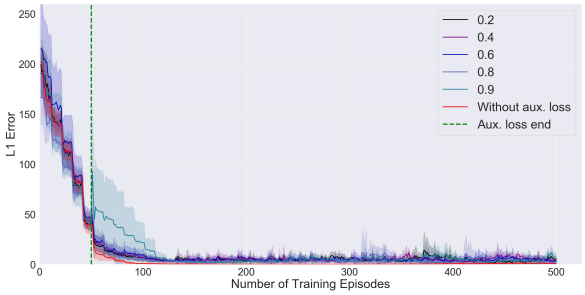


$target\_update = 10$. $target\_update = 100$.

(a) Duration: 5 episodes.

Figure A.37: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{128 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.

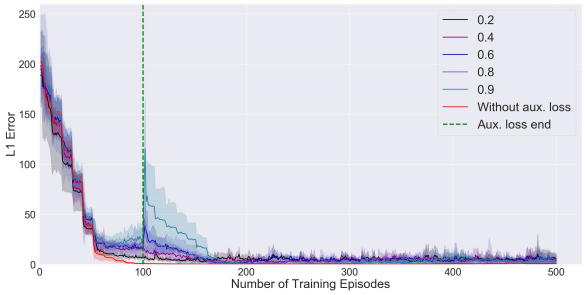
[9]Notice that this is different for the FrozenLake 4x4 (OH) domain, where 2-layer DQNs trained without auxiliary loss converge to the true Q-values much after the target network has been updated a sufficient number of times.



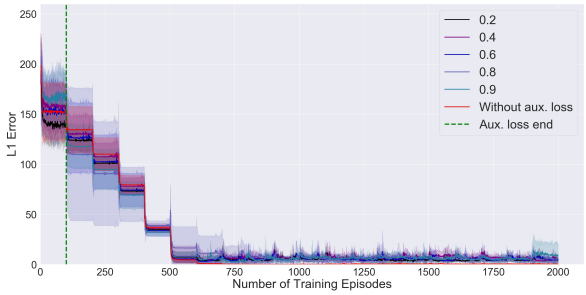
$target\_update = 10$.

$target\_update = 100$.

(b) Duration: 50 episodes.



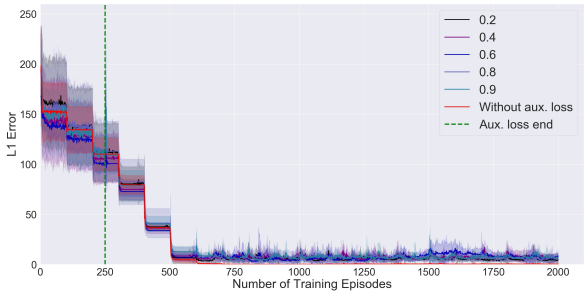
$target\_update = 10$.



$target\_update = 100$.

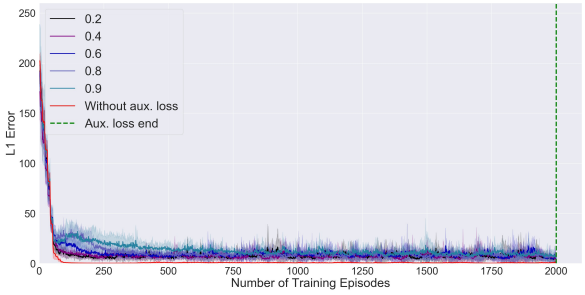
(c) Duration: 100 episodes.

$target\_update = 10$.

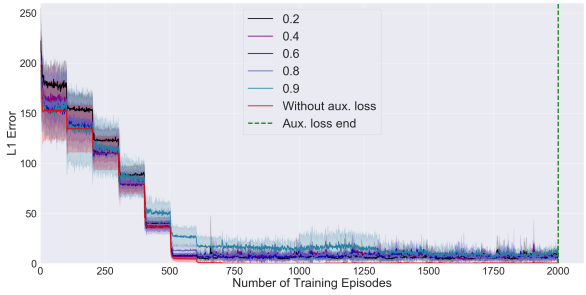


$target\_update = 100$.

(d) Duration: 250 episodes.



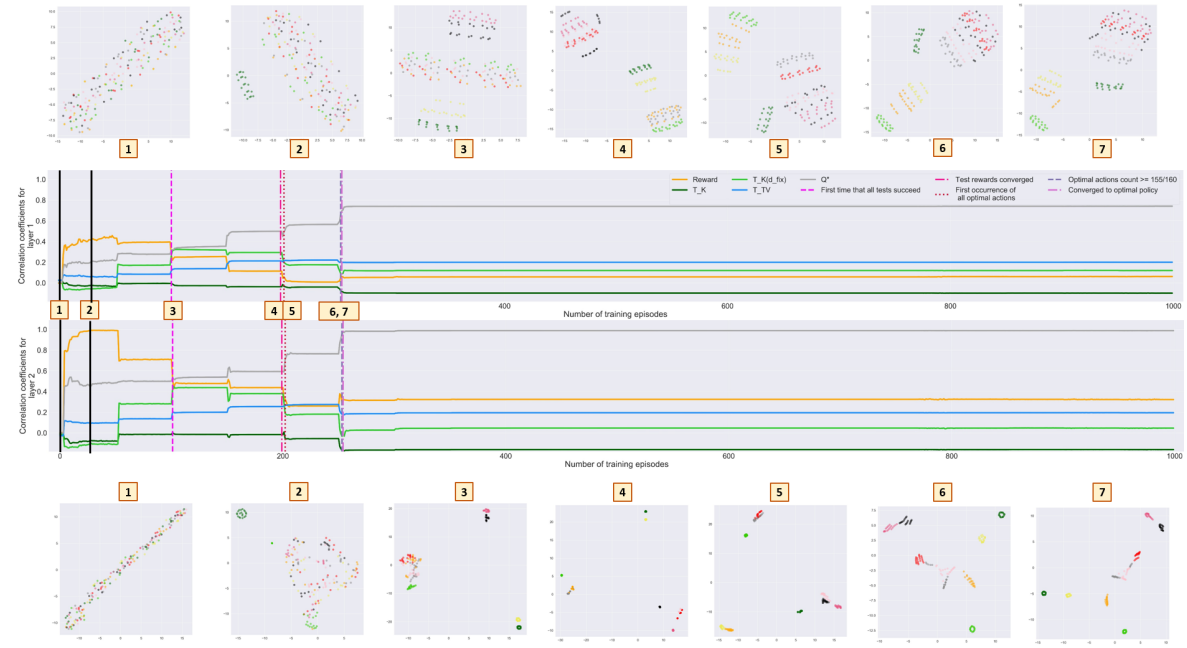
$target\_update = 10$.



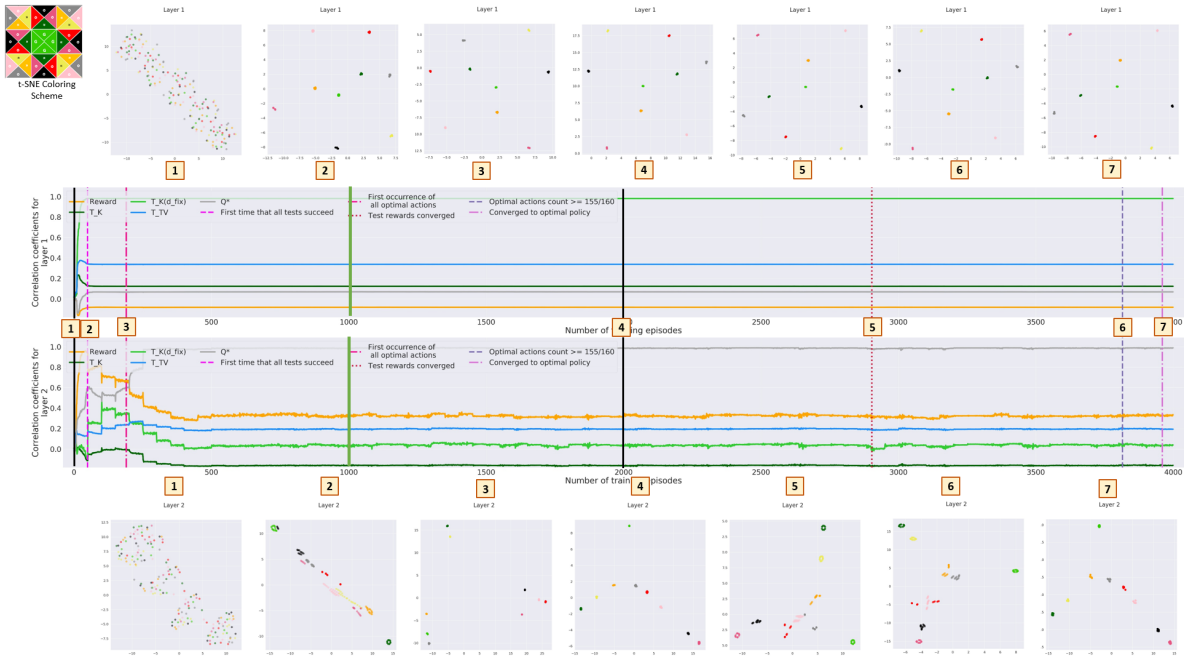
$target\_update = 100$.

(e) Duration: $2,000$ episodes.

Figure A.37: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{128 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.

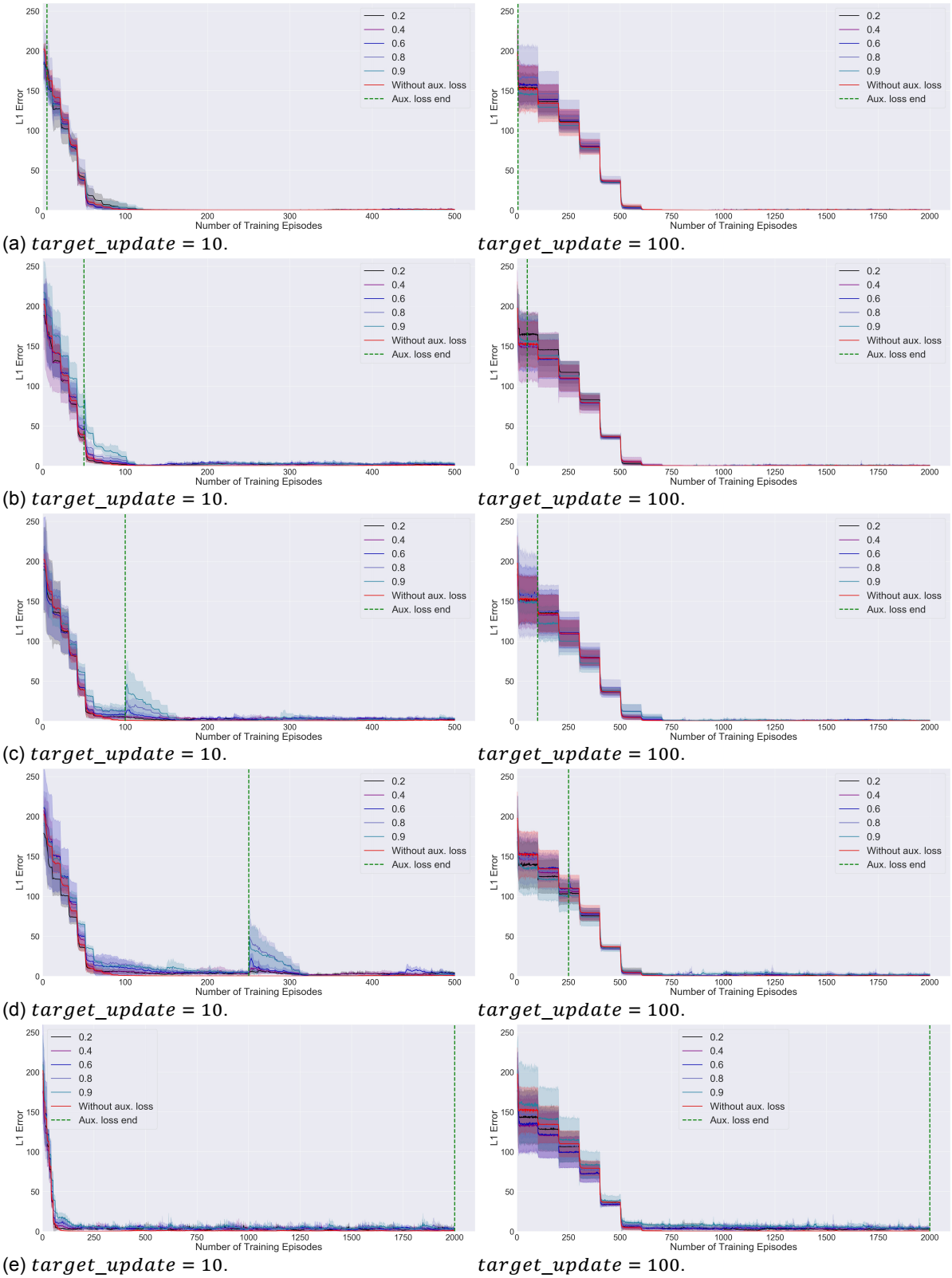


(a) Without auxiliary loss.



(b) With auxiliary loss.

Figure A.38: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 10 for the Gridworld 3x3 (OH) domain with and without the auxiliary loss added to the training. d_{fix} is employed as bisimulation-based state distance for the auxiliary loss with $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first $1,000$ training episodes, the latter of which is indicated by the green vertical line.



(a) $target\_update = 10$. $target\_update = 100$.

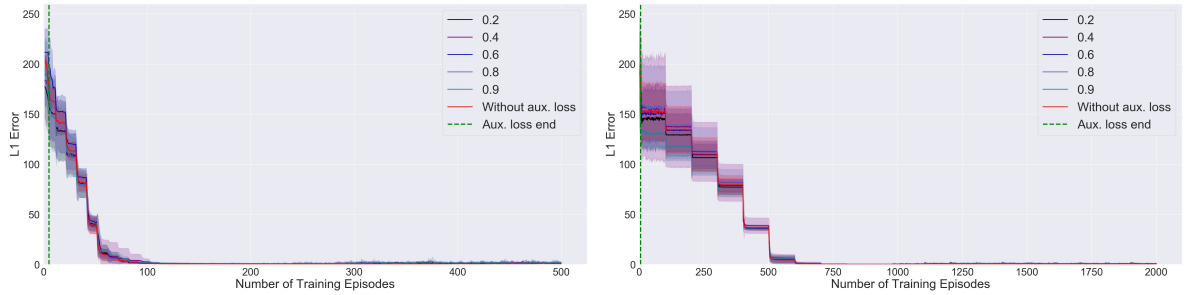
(b) $target\_update = 10$. $target\_update = 100$.

(c) $target\_update = 10$. $target\_update = 100$.

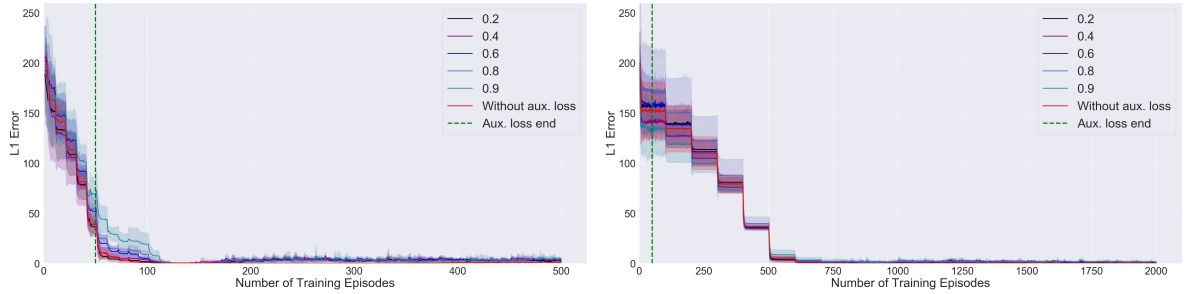
(d) $target\_update = 10$. $target\_update = 100$.

(e) $target\_update = 10$. $target\_update = 100$.

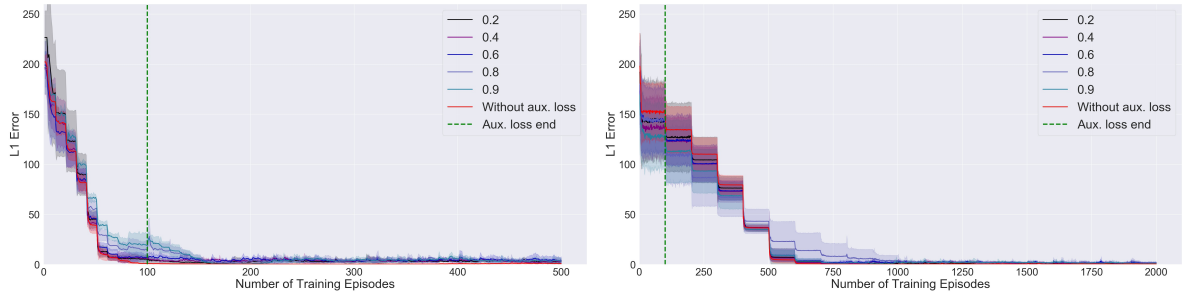
Figure A.39: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.



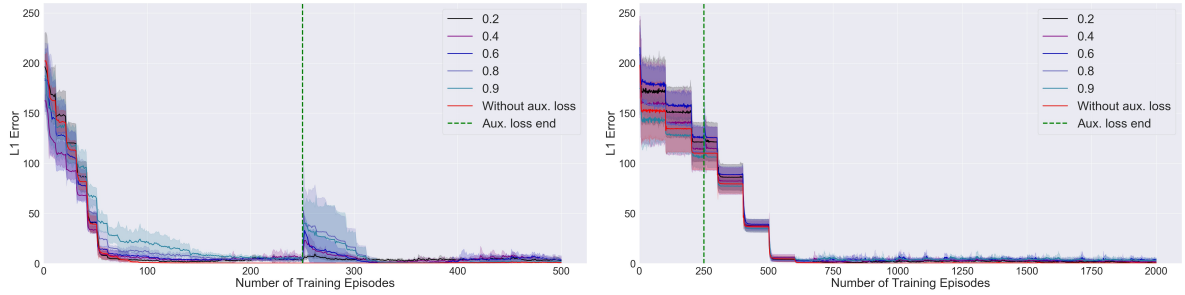
(a) $target\_update = 10$. $target\_update = 100$.



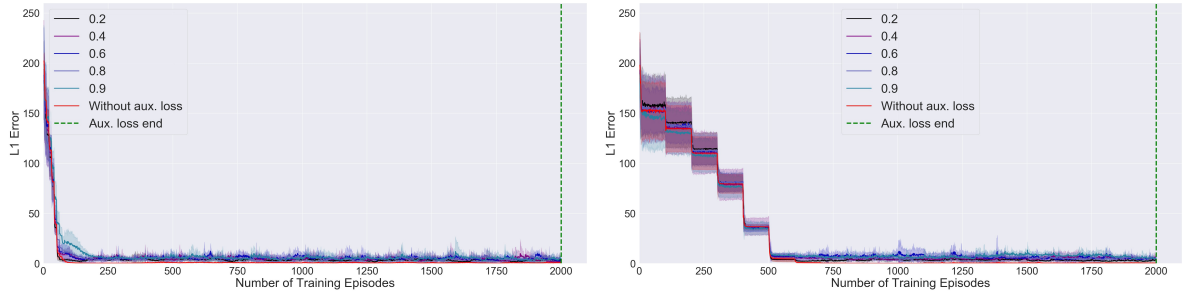
(b) $target\_update = 10$. $target\_update = 100$.



(c) $target\_update = 10$. $target\_update = 100$.



(d) $target\_update = 10$. $target\_update = 100$.



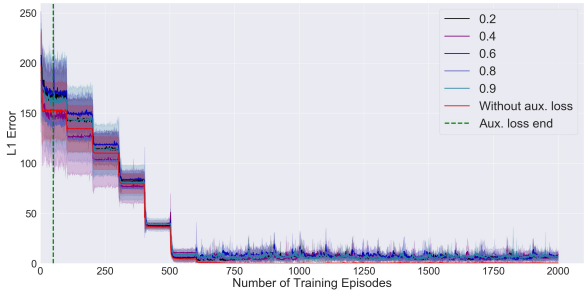
(e) $target\_update = 10$. $target\_update = 100$.

Figure A.40: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{8 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.

(a) $target\_update = 10$.

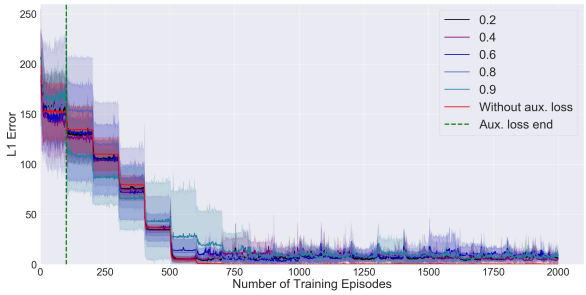
$target\_update = 100$.

(b) $target\_update = 10$.

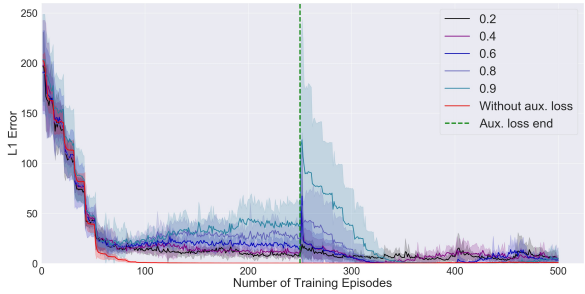


$target\_update = 100$.

(c) $target\_update = 10$.

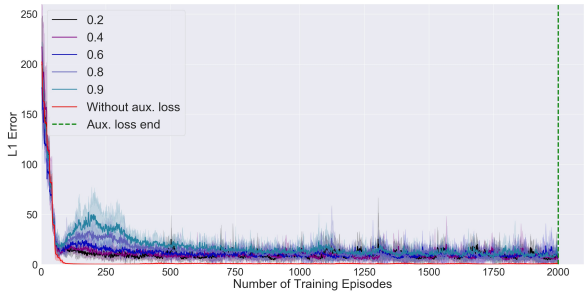


$target\_update = 100$.

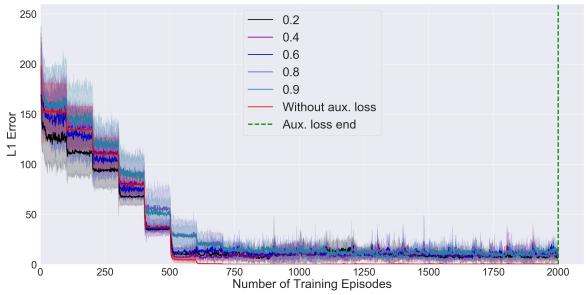


(d) $target\_update = 10$.

$target\_update = 100$.

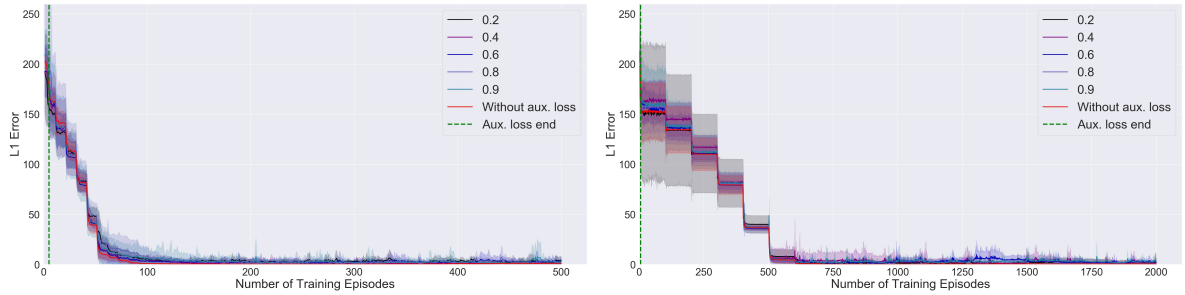


(e) $target\_update = 10$.

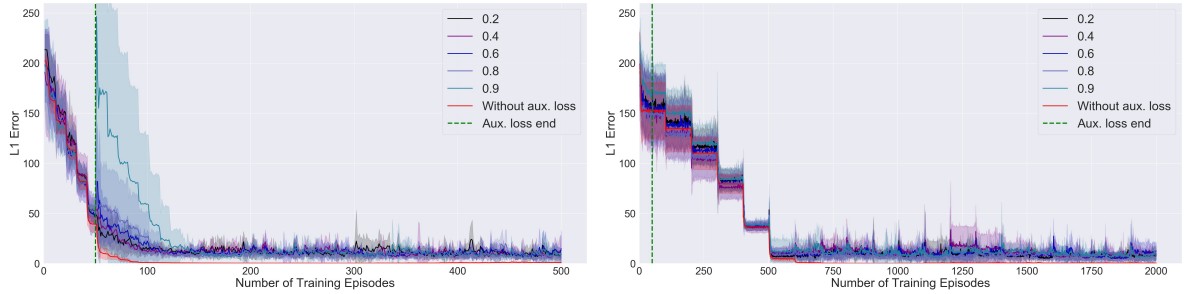


$target\_update = 100$.

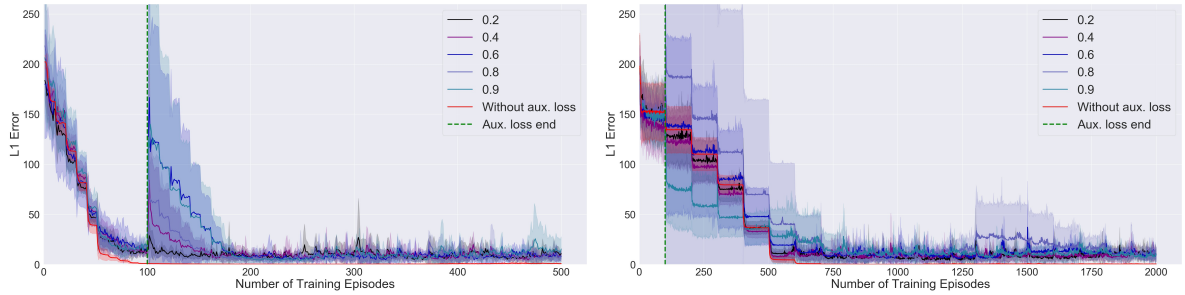
Figure A.41: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{256 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.



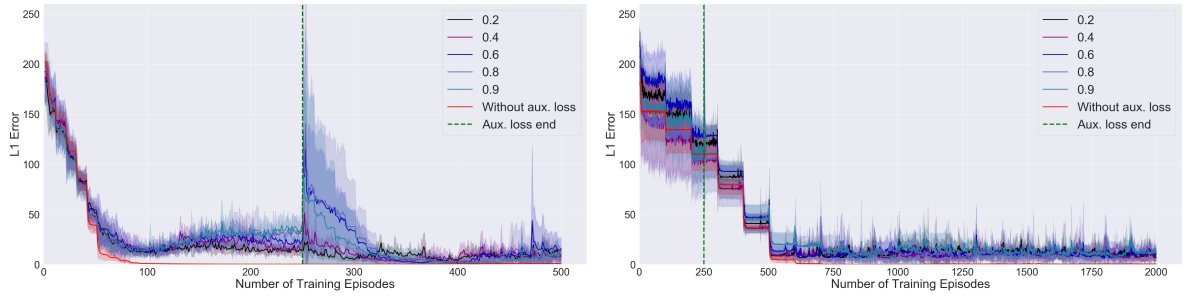
(a) $target\_update = 10$. $target\_update = 100$.



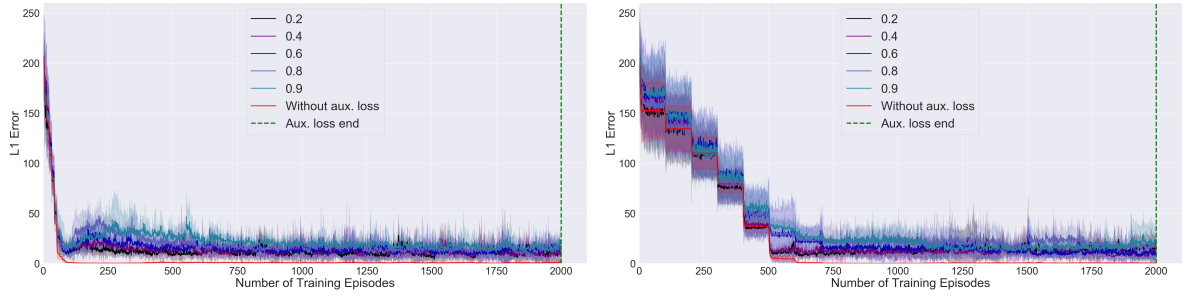
(b) $target\_update = 10$. $target\_update = 100$.



(c) $target\_update = 10$. $target\_update = 100$.



(d) $target\_update = 10$. $target\_update = 100$.



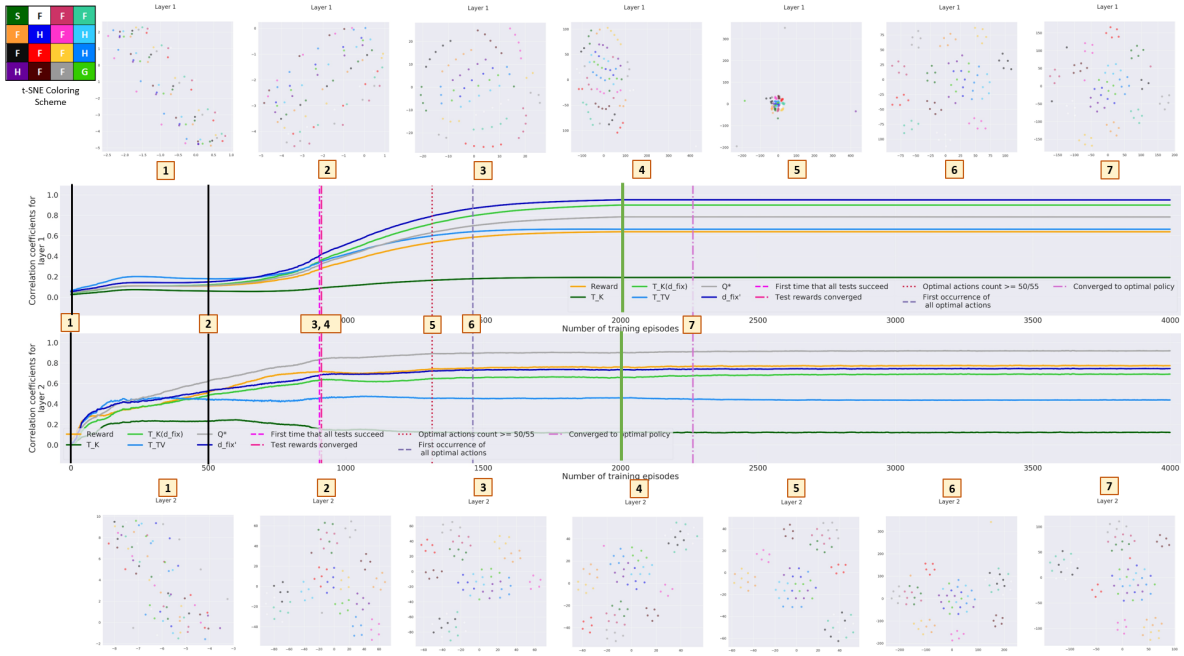
(e) $target\_update = 10$. $target\_update = 100$.

Figure A.42: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the Gridworld 3x3 (OH) domain when employing different weights for the auxiliary loss, different target network update frequencies, and terminating the usage of the auxiliary loss after different numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{512 \times hidden\_layer\_size}$ and d_{fix} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown. For $target\_update = 10$, some figures show the L_1-error during fewer training episodes to make differences more visible.

A.2.3. Auxiliary Loss Based on d'_{fix}

The impact of using an auxiliary loss based on d'_{fix} is very similar to employing one that is based on d_{fix}, if one adjusts d_E^{max} so that the target Euclidean distances of the activations of states with the maximum occurring bisimulation-based distance are comparable[10]. Comparing Figure A.43a to Figure A.43b, one can see that if the values for d_E^{max} are set to comparable values, employing d'_{fix} and d_{fix} leads to very similar first-layer state representations. For example, $c_{K(d_{fix})}$ takes on a value near 0.9 in the first layer for training episode $2,000$ in both cases, with the value being only slightly higher when d_{fix} is utilized.

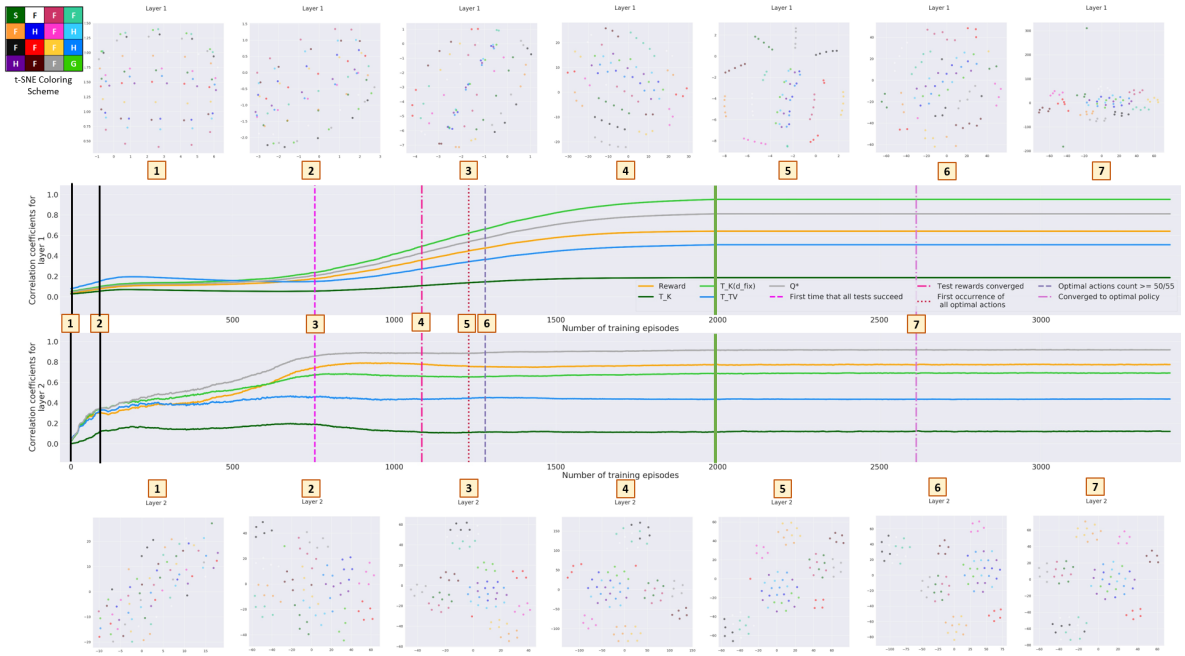
It is therefore not surprising that the L_1-errors with respect to the true Q-values during training are also comparable for several weights and durations for the auxiliary loss as visualized in Figure A.44. Yet, whereas using an auxiliary loss based on d_{fix} not only allows for lower L_1-errors at the beginning of training than when no auxiliary loss is utilized, but also for significantly earlier convergence to the true Q-values if the auxiliary loss is applied for $1,500$ or $2,000$ training episodes, the latter is not the case when d'_{fix} is employed (see Figures A.44c and A.44d). The likely reason is that d'_{fix} in contrast to d_{fix} does not assign distances of exactly 0 to bisimilar states. This makes it more difficult to learn not to distinguish states based on the superfluous feature value, as well as to group bisimilar states from different ground states together while the auxiliary loss is applied. An auxiliary loss based on d'_{fix} should hence be terminated sooner or used with a lower weight than one based on d_{fix}, so that the learning of precise Q-values is not hindered towards the end of training.



(a) d'_{fix}. $d_E^{max} = \sqrt{0.426 \times hidden\_layer\_size}$, which is comparable to $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$ when using d_{fix}.

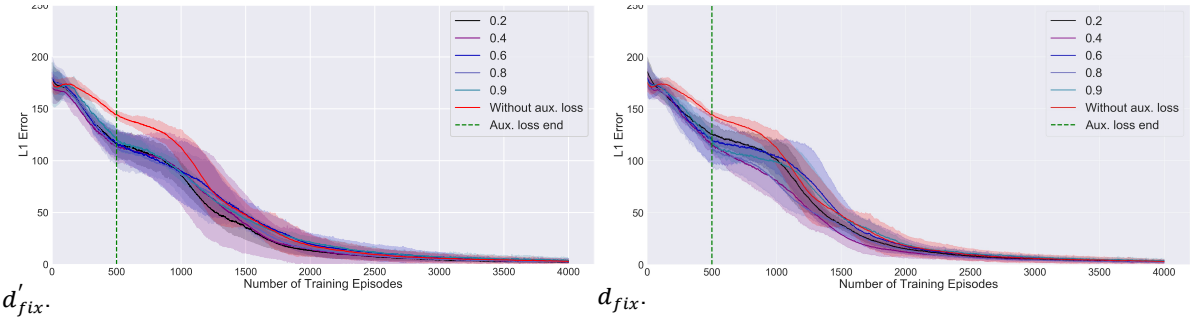
Figure A.43: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when using either d'_{fix} or d_{fix} as basis of the auxiliary loss. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first $2,000$ training episodes, the latter of which is indicated by the green vertical line. Notice the slightly different x-axis ranges of a) and b).

[10] The highest occurring value for FrozenLake 4x4 for d_{fix} is 0.2, whereas the one for d'_{fix} is 4.891. This means that the two values differ by a factor of roughly 24.5.



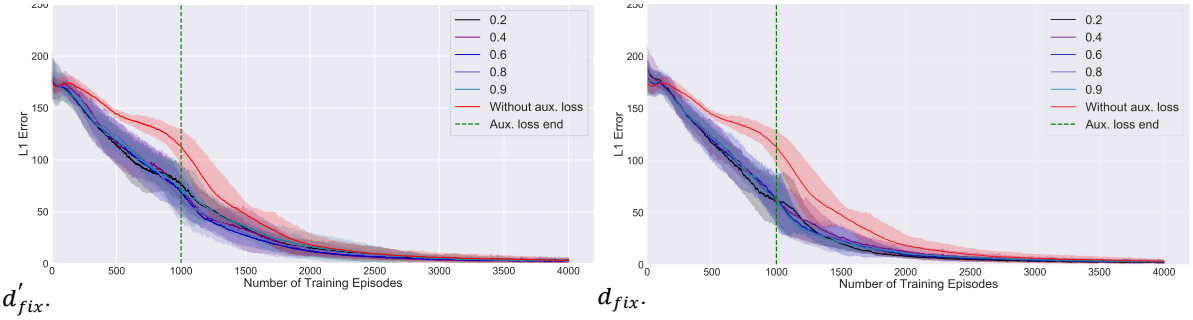
(b) d_{fix}. $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$.

Figure A.43: Correlation coefficients and t-SNE plots of the activations the states are mapped to for the layers of a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when using either d'_{fix} or d_{fix} as basis of the auxiliary loss. The auxiliary loss is applied with a weight of 0.2 and a decay rate of 0.9999 during the first $2,000$ training episodes, the latter of which is indicated by the green vertical line. Notice the slightly different x-axis ranges of a) and b).



d'_{fix}. d_{fix}.

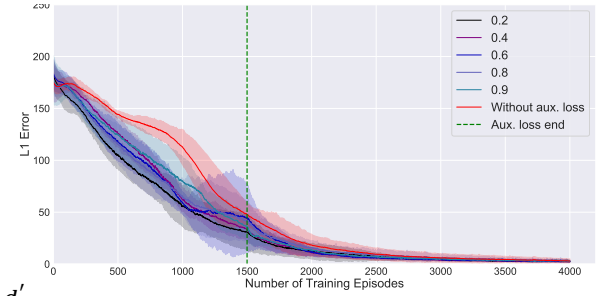
(a) Duration: 500 episodes.



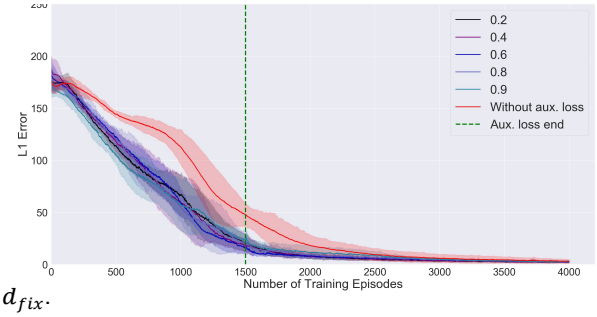
d'_{fix}. d_{fix}.

(b) Duration: $1,000$ episodes.

Figure A.44: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss and $target\_update = 5$. The usage of the auxiliary loss is ended after varying numbers of training episodes, which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{0.426 \times hidden\_layer\_size}$ for d'_{fix} and to $\sqrt{256 \times hidden\_layer\_size}$ for d_{fix}. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

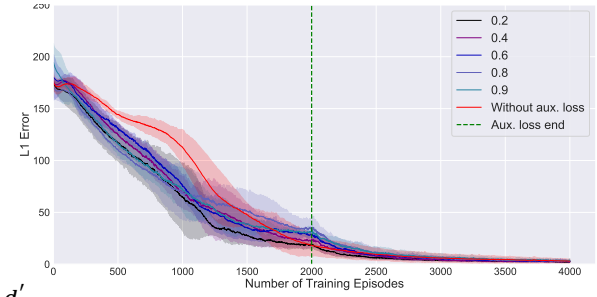


d'_{fix}.



d_{fix}.

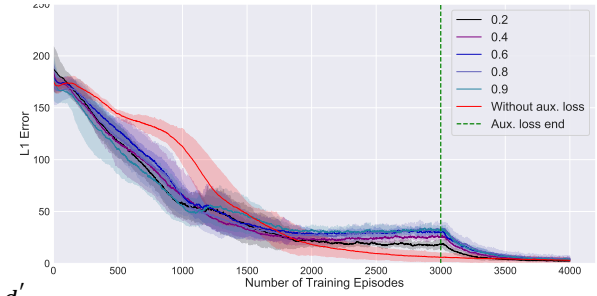
(c) Duration: $1,500$ episodes.



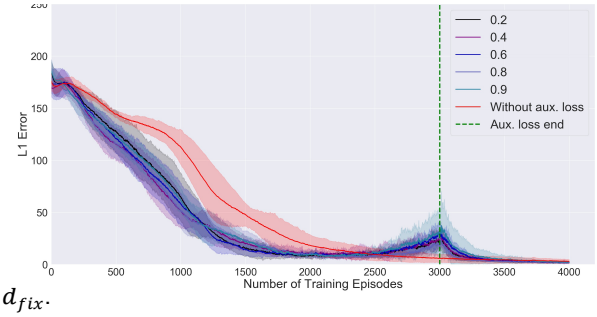
d'_{fix}.

d_{fix}.

(d) Duration: $2,000$ episodes.



d'_{fix}.



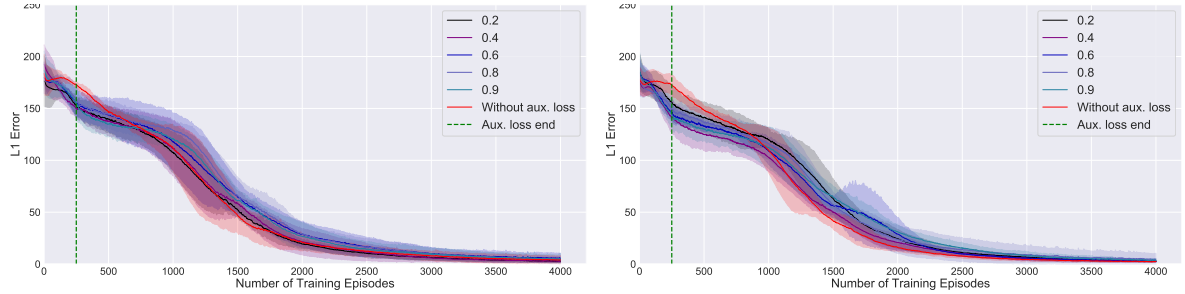
d_{fix}.

(e) Duration: $3,000$ episodes.

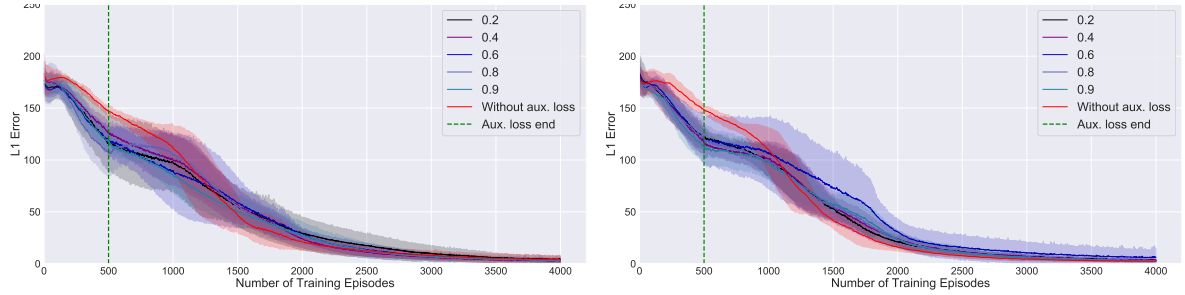
Figure A.44: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss and $target\_update = 5$. The usage of the auxiliary loss is ended after varying numbers of training episodes, which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{0.426 \times hidden\_layer\_size}$ for d'_{fix} and to $\sqrt{256 \times hidden\_layer\_size}$ for d_{fix}. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

A.2.4. Auxiliary Loss Based on T_{TV}

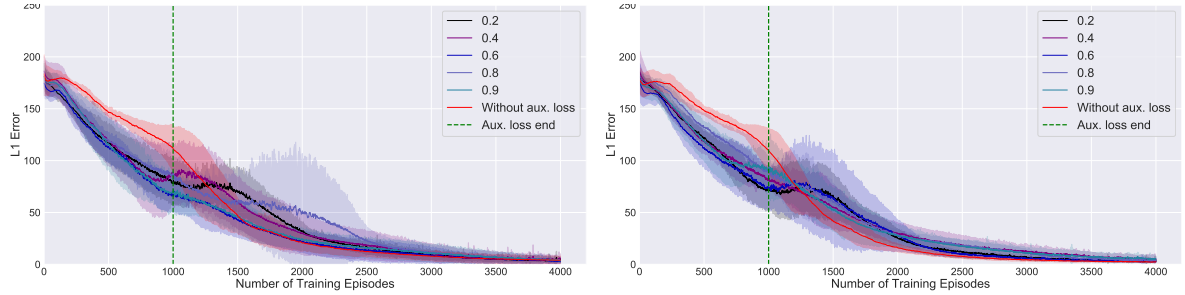
If an auxiliary loss based on T_{TV} is introduced to the training of 2-layer DQNs for the FrozenLake 4x4 (OH) domain, the L_1-error with respect to the Q-values is improved at the beginning of training (see Figure A.45). Yet, similarly to employing an auxiliary loss based on d'_{fix}, the addition of this auxiliary loss does not allow DQNs to converge sooner than if no auxiliary loss is utilized. Even though we performed experiments solely with one value for d_E^{max} and hence do not know for sure whether different values would enable earlier convergence, it makes sense that learning is improved upon only at the beginning of training. This is the case, because grouping states together based on T_{TV} leads to distinct clusters for bisimilar states from different ground states, such as the terminal states in the FrozenLake 4x4 domain. Consequently, while the activations of states that differ solely in the superfluous feature value are pushed closer together by means of this auxiliary loss, which explains the improvement in L_1-error at the beginning of training, the activations of states belonging to different ground states are pulled farther apart by the auxiliary loss, thus making it more difficult to learn to map such states to the same Q-values in the second layer. Therefore, in practice, this auxiliary loss should also be applied solely at the very beginning of training or its weight should be decayed quickly.



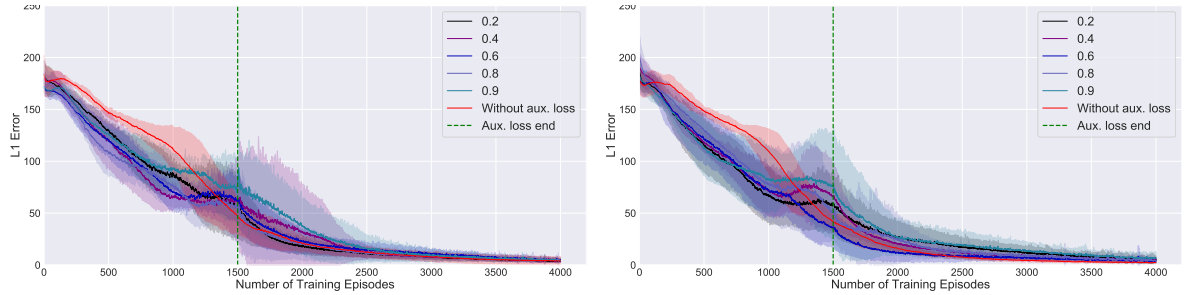
(a) $target\_update = 1$. $target\_update = 4$.



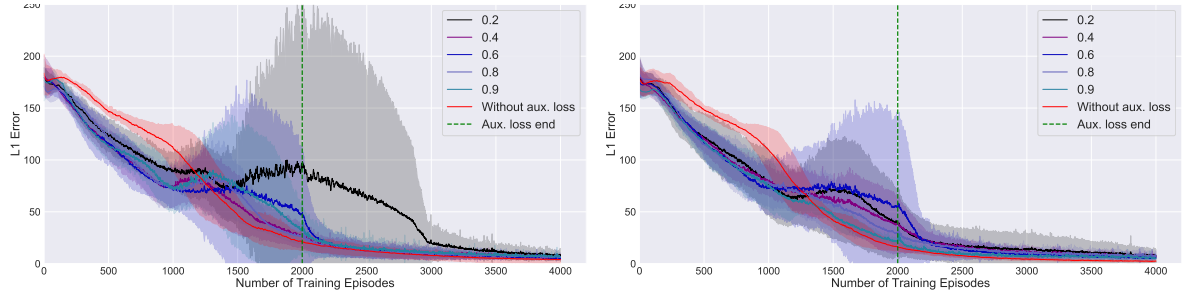
(b) $target\_update = 1$. $target\_update = 4$.



(c) $target\_update = 1$. $target\_update = 4$.



(d) $target\_update = 1$. $target\_update = 4$.



(e) $target\_update = 1$. $target\_update = 4$.

Figure A.45: L_1-error with respect to the true Q-values for a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain when employing different weights for the auxiliary loss and different target network update frequencies, and terminating the usage of the auxiliary loss after varying numbers of training episodes, the latter of which is indicated by the green vertical line. d_E^{max} is set to $\sqrt{128 \times hidden\_layer\_size}$ and T_{TV} is employed as bisimulation-based state distance for the auxiliary loss. The red line depicts the L_1-error when no auxiliary loss is used. 95%-confidence intervals are shown based on 5 repetitions.

A.3. Impact of Markovianity on Generalization

Section A.3.1 contains further information on generalization to related domains with modified reward or transition functions, and Section A.3.2 delineates in more detail our methodology and results regarding the generalization to new irrelevant feature values.

A.3.1. Transfer to Related Domains

Section A.3.1.1 outlines methodological details, Section A.3.1.2 analyzes the impact of changing the Euclidean distance between the first-layer activations of non-bisimilar states on the generalization performance, and Section A.3.1.3 provides detailed results for the experiments conducted for the domains related to the FrozenLake 4x4 and Gridworld 3x3 domains.

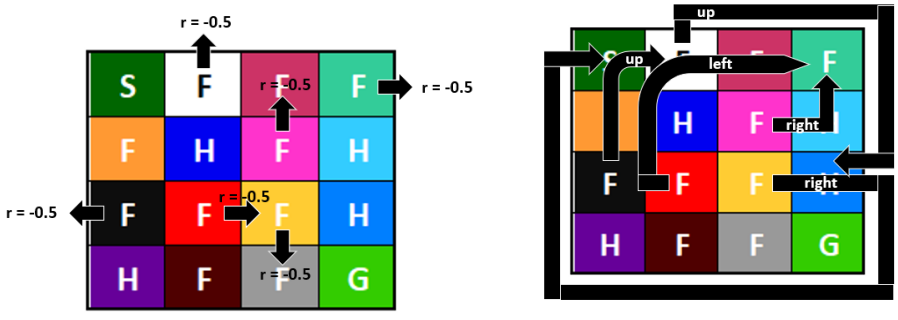
A.3.1.1 Methodological Details

Domains related to FrozenLake 4x4. We design two domains related to the FrozenLake 4x4 domain, in each of which two states are bisimilar if and only if they are bisimilar in the original domain:

1. *Modified reward function*. We change the reward function by setting the immediate reward for going up in ground state 1, right in ground state 3, up in ground state 6, left in ground state 8, right in ground state 9 and down in ground state 10 to -0.5 instead of the original value of 0. This change causes different optimal actions for ground states 9 and 10 and different Q-values for those two ground states and all states for which the path under the optimal policy leads to one of the two ground states for some action. This related domain is shown in Figure A.46a and denoted by FrozenLake^R 4x4.

2. *Modified transition and reward functions*. We replace the original deterministic transition function by a new deterministic transition function, in which the next ground states for five state-action combinations are altered (see Figure A.46b). Thereby, the new transition probabilities ensure that each ground state is still visited for some state-action combination. The change in transition probabilities leads to different Q-values for some states as well as different immediate rewards due to the fact that stepping into a hole yields a reward of -1. Yet, the optimal actions of all states are identical to the ones in the original domain.

The results for the experiments based on these domains serve as supporting evidence and can be found in Section A.3.1.3.



(a) FrozenLake^R 4x4: Modified reward function.

(b) FrozenLake^{RT} 4x4: Modified reward and transition functions.

Figure A.46: Domains related to the FrozenLake 4x4 domain. The coloring of the ground states is based on the colors we assign to the activations of states in our t-SNE plots. Black arrows indicate changes in the reward or transition function compared to the original domain. If a modification is related to the transition function, the black arrow is labeled with the action for which the next state is altered, and if a modification occurs with respect to the reward function, the black arrow is labeled with the new immediate reward r.

Auxiliary loss. For some experiments, we apply the auxiliary loss proposed in Chapter 4 based on d_{fix} during the first part of the training on the original domain for the three types of state encoding in which the ground state is one-hot encoded. Since the first network layer cannot form the coarsest Markov state representation for the (F) state encoding, as it requires learning a non-linearly separable

function, the auxiliary loss is not applied for this form of state encoding. The specific hyperparameter values are named in Table A.1 and Table A.2 for the Gridworld 3x3 and the FrozenLake 4x4 domain, respectively. Smaller values for d_E^{max} and lower weights for the auxiliary loss are necessary for very small hidden layer sizes to keep the gradients from exploding during training. Moreover, since an abstraction coarser than the coarsest Markov state representation needs to be formed in the first layer for very small hidden layer sizes such as 3 for the DQN to be able to converge to the true Q-values, the auxiliary loss has to be terminated sooner for such hidden layer sizes. Yet, despite using adapted settings for DQNs with smaller hidden layer sizes, adding the bisimulation-based auxiliary loss to the training of DQNs with a hidden layer size of 3 does not always prevent the gradients from exploding. To render the results more comparable, we hence discard any repetition for this hidden layer size from our results for which training a DQN leads to exploding gradients when the auxiliary loss is used.

Table A.1: Hyperparameter values for the bisimulation-based auxiliary loss that is added to the training of 2-layer DQNs for the Gridworld 3x3 domain in the context of transfer to related domains. Values depend on the hidden layer size (HLS) of a DQN.

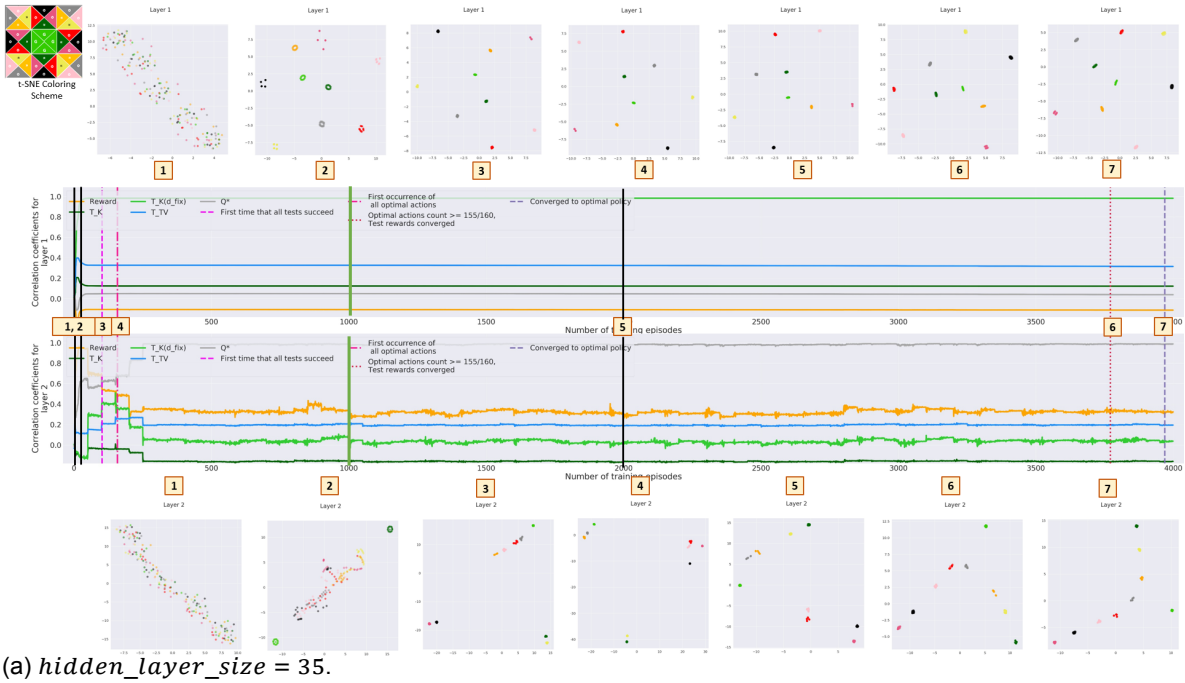
Parameter	Description	HLS 3	HLS 5	HLS > 5
d_E^{max}	Target Euclidean distance of the activations of two states with the max. bisimulation-based distance.	$\sqrt{32 \times HLS}$	$\sqrt{32 \times HLS}$	$\sqrt{128 \times HLS}$
Decay rate	Decay rate, applied after each application of the auxiliary loss.	0.9999	0.9999	0.9999
Measure	Bisimulation-based measure used for the auxiliary loss.	d_{fix}	d_{fix}	d_{fix}
Stop episode	Training episode starting at which the auxiliary loss is no longer applied.	500	500	1,000
Weight	Weight of the auxiliary loss.	0.0001	0.1	0.2

Table A.2: Hyperparameter values for the bisimulation-based auxiliary loss that is added to the training of 2-layer DQNs for the FrozenLake 4x4 domain in the context of transfer to related domains. Values depend on the hidden layer size (HLS) of a DQN.

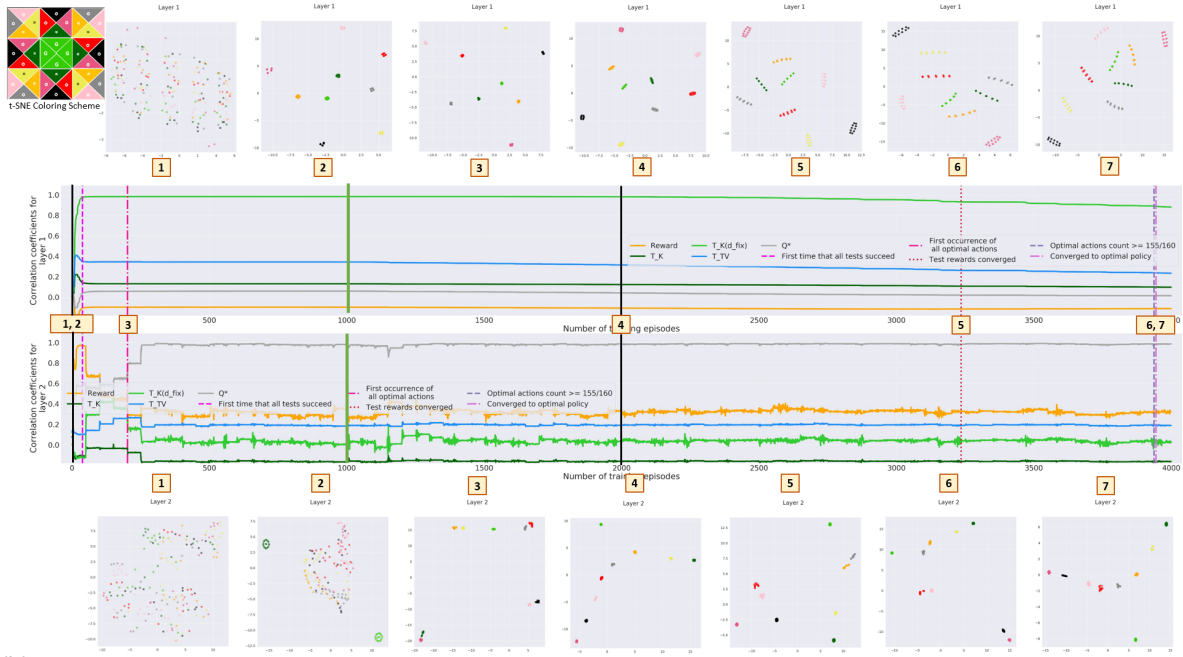
Parameter	Description	HLS $= 3$	HLS > 3
d_E^{max}	Target Euclidean distance of the activations of two states with the max. bisimulation-based distance.	$\sqrt{8 \times HLS}$	$\sqrt{8 \times HLS}$
Decay rate	Decay rate, applied after each application of the auxiliary loss.	0.9999	0.9999
Measure	Bisimulation-based measure used for the auxiliary loss.	d_{fix}	d_{fix}
Stop episode	Training episode starting at which the auxiliary loss is no longer applied.	2,000	2,000
Weight	Weight of the auxiliary loss.	0.01	0.1

While we do adjust the hyperparameter values to account for the fact that small hidden layer sizes imply that a first-layer representation coarser than the coarsest Markov state representation needs to be learned to allow for convergence to the true Q-values, we do not fine-tune the values to ensure that all larger-than-necessary hidden layer sizes learn first-layer representations that are equally Markov. For

example, if the auxiliary loss is applied for solely $1,000$ out of $4,000$ training episodes for the Gridworld 3x3 domain, the first-layer representation tends to become slightly less Markov again at the very end of training for large hidden layer sizes such as 65 (see Figure A.47). This aspect is important to keep in mind when using our results to compare the generalization performances of 2-layer DQNs with different hidden layer sizes when they are trained with the bisimulation-based auxiliary loss.



(a) $hidden\_layer\_size = 35$.



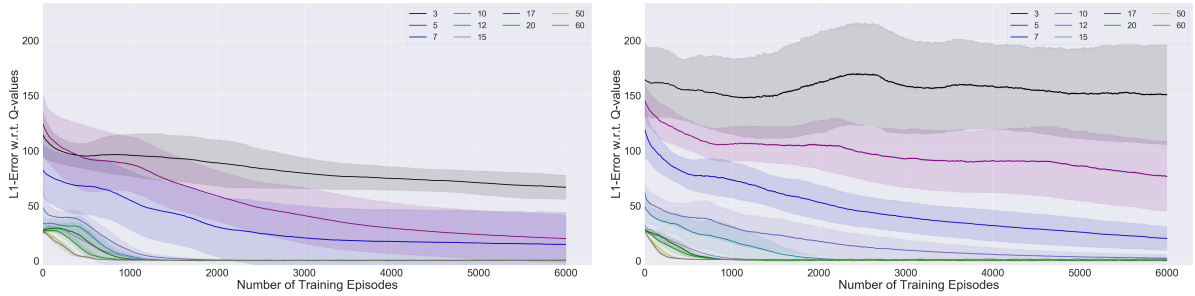
(b) $hidden\_layer\_size = 65$.

Figure A.47: Correlation coefficients and t-SNE plots of the activations states are mapped to in the layers of 2-layer DQNs with different hidden layer sizes for the Gridworld 3x3 (OH) domain during training. The auxiliary loss is applied with a weight of 0.2 during the first $1,000$ training episodes, the latter of which is indicated by the green vertical line.

A.3.1.2 Impact of d_E^{max}

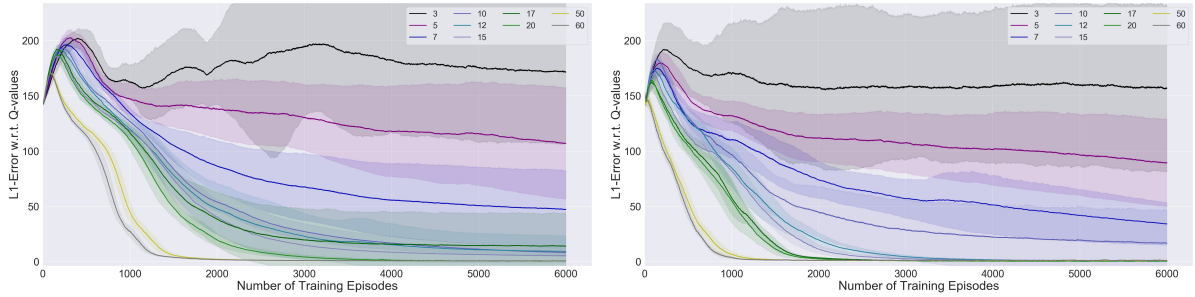
Figure A.48 and Figure A.49 show for the FrozenLake^R 4x4 and the FrozenLake^{RT} 4x4 domain, respectively, that increasing the value for d_E^{max} for the auxiliary loss during training on the original domain

generally leads to earlier convergence to the true Q-values by DQNs with large hidden layers. Since d_E^{max} determines the target Euclidean distances between the activations of non-bisimilar states, larger Euclidean distances between the activations of non-bisimilar states hence lead to improved generalization performance. Yet, generalization performance tends to be slightly worse for DQNs with small hidden layer sizes such as 3 and 5 when d_E^{max} is increased. This deterioration can be explained by two aspects. First, a higher value for d_E^{max} makes it more difficult for DQNs with small hidden layers to converge to the true Q-values on the original domain. This results in a higher L_1-error at the beginning of retraining when the output layer's weights are transferred, which causes less accurate Q-values to be learned in the corresponding two transfer scenarios due to the less favorable initialization. Second, the larger Euclidean distances between non-bisimilar states in the first layer themselves also make it harder for DQNs with small hidden layers to learn the Q-values on the modified domain. The reason is that even when the output layer's weights are not transferred, DQNs with small hidden layers tend to generalize worse for higher values for d_E^{max} (see Figure A.49b).



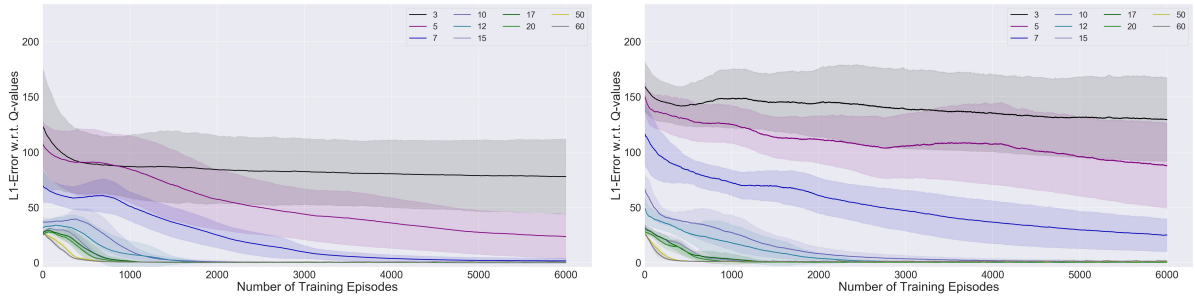
$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$

(a) FrozenLake^R 4x4: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$

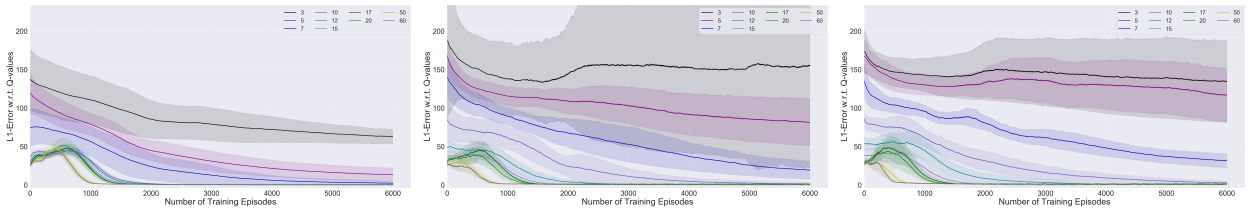
(b) FrozenLake^R 4x4: The weights from solely the first network layer are transferred and those are not updated during retraining.



$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$

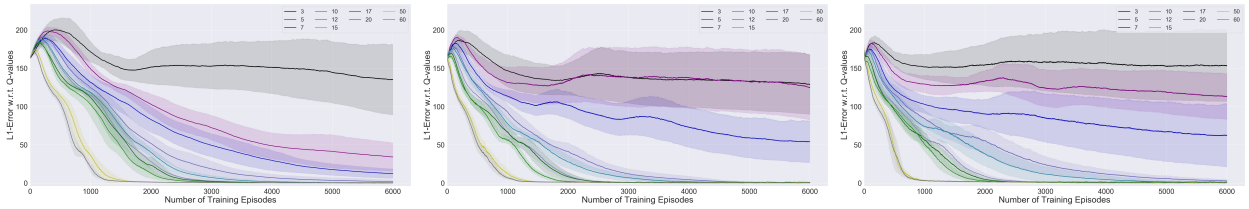
(c) FrozenLake^R 4x4: The weights from both network layers are transferred and updated during retraining.

Figure A.48: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the FrozenLake^R 4x4 (OH) domain for each of the three forms of transfer when using different values for d_E^{max} when applying the auxiliary loss during training on the original domain. Values are based on 10 repetitions. 95%-confidence intervals are shown.



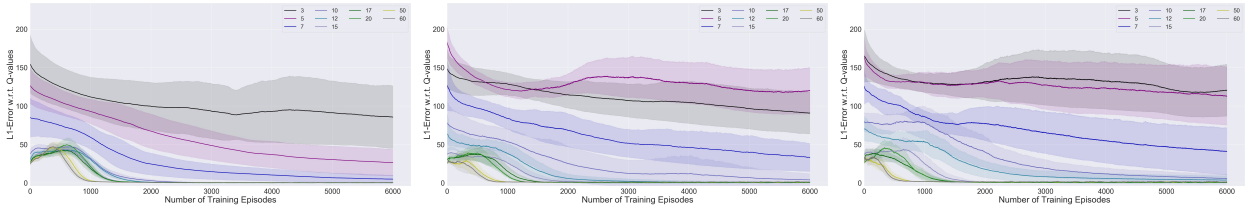
$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$

(a) FrozenLake^{RT} 4x4: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$

(b) FrozenLake^{RT} 4x4: The weights from solely the first network layer are transferred and those are not updated during retraining.



$d_E^{max} = \sqrt{8 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{128 \times hidden\_layer\_size}$ $d_E^{max} = \sqrt{256 \times hidden\_layer\_size}$

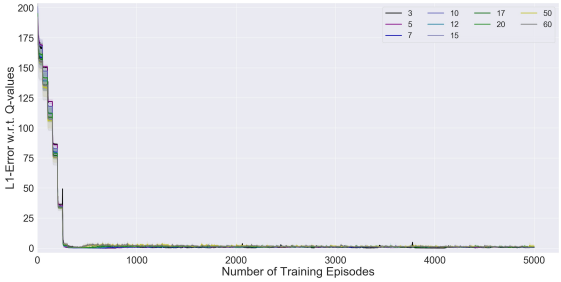
(c) FrozenLake^{RT} 4x4: The weights from both network layers are transferred and updated during retraining.

Figure A.49: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the FrozenLake^{RT} 4x4 (OH) domain for each of the three forms of transfer when using different settings for d_E^{max} when applying the auxiliary loss during training on the original domain. Values are based on 10 repetitions. 95%-confidence intervals are shown.

A.3.1.3 Detailed Results

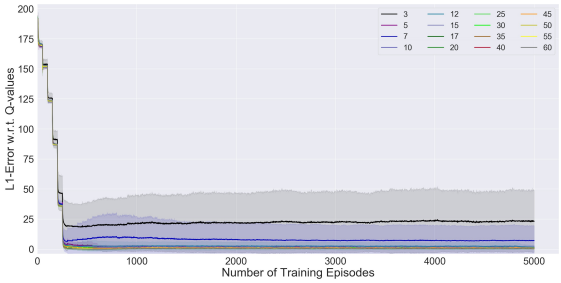
Figures A.50 and A.51 and Figure A.52 provide detailed results for the transfer to related domains for the Gridworld 3x3 and the FrozenLake 4x4 domain, respectively. Results are shown for all transfer modes and types of state encoding and for pretraining both with and without the bisimulation-based auxiliary loss.

Without bisimulation-based auxiliary loss.

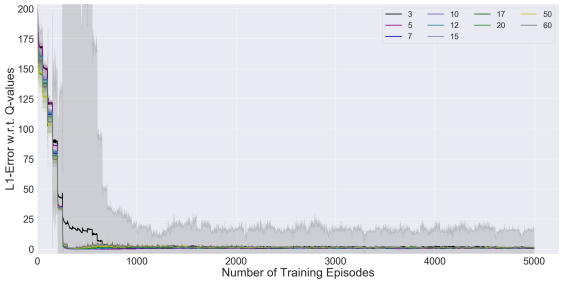


With bisimulation-based auxiliary loss.

(a) Gridworld 3x3 (OH).

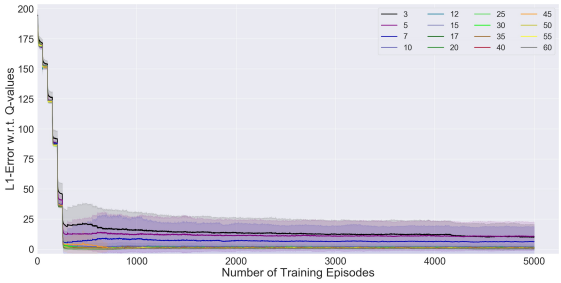


Without bisimulation-based auxiliary loss.

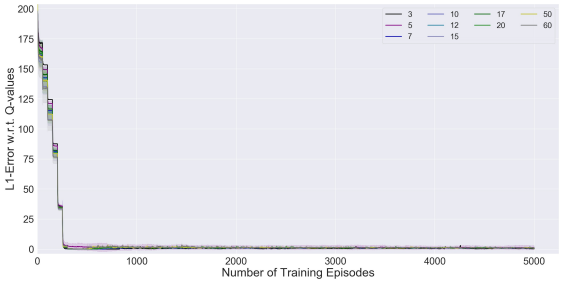


With bisimulation-based auxiliary loss.

(b) Gridworld 3x3 (OH)(N).

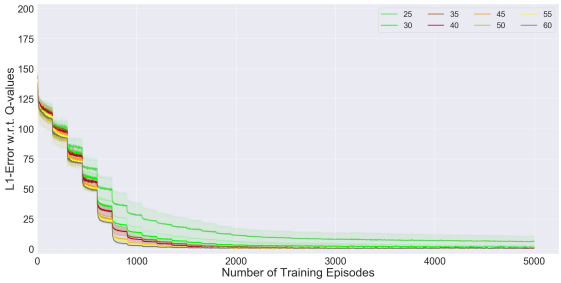


Without bisimulation-based auxiliary loss.



With bisimulation-based auxiliary loss.

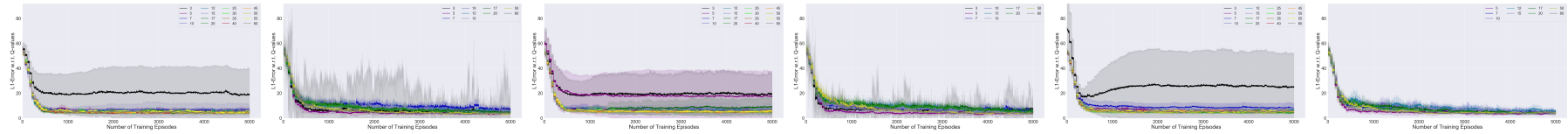
(c) Gridworld 3x3 (OH)(F-OH).



Without bisimulation-based auxiliary loss.

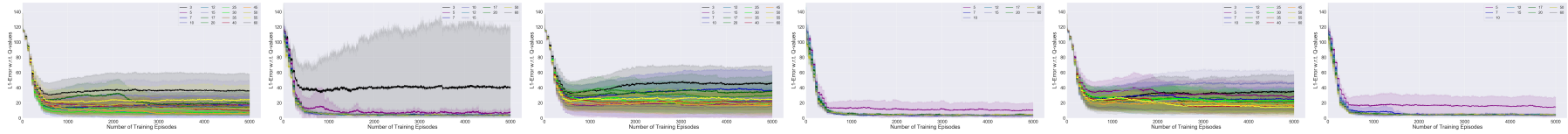
(d) Gridworld 3x3 (F).

Figure A.50: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the Gridworld^R 3x3 domain. The first-layer weights are initialized to those of DQNs trained on the Gridworld 3x3 domain either with or without auxiliary loss and are not updated during retraining. The second-layer weights are newly initialized. Values are based on 10 repetitions and 95%-confidence intervals are shown.



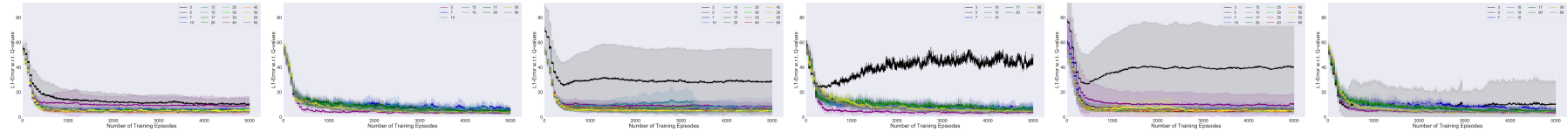
(OH). (OH)(N). (OH)(F-OH).

(a) Gridworld^T 3x3: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



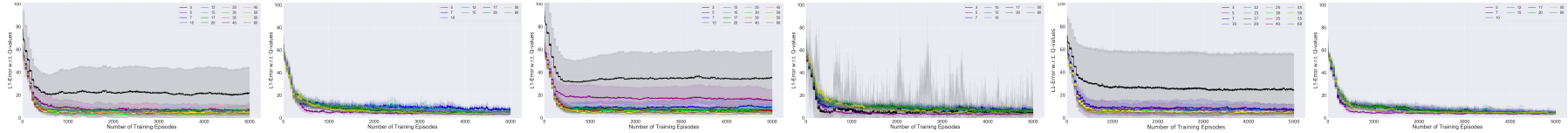
(OH). (OH)(N). (OH)(F-OH).

(b) Gridworld^T 3x3: The weights from solely the first network layer are transferred and those are not updated during retraining.



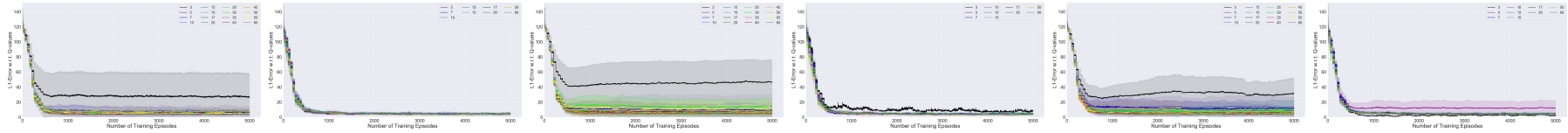
(OH). (OH)(N). (OH)(F-OH).

(c) Gridworld^T 3x3: The weights from both network layers are transferred and updated during retraining.



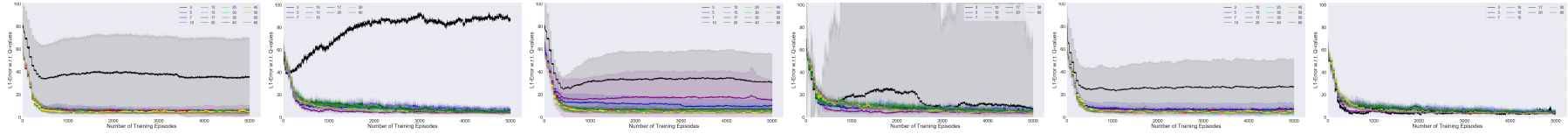
(OH). (OH)(N). (OH)(F-OH).

(d) Gridworld^{RT} 3x3: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



(OH). (OH)(N). (OH)(F-OH).

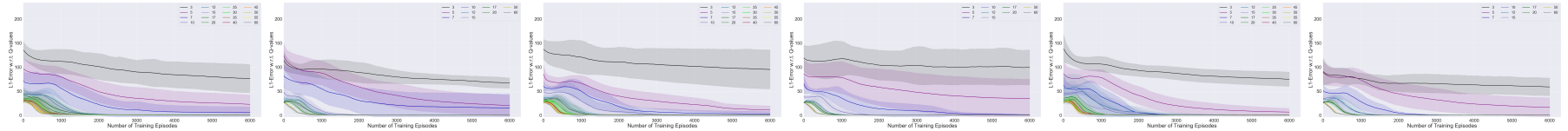
(e) Gridworld^{RT} 3x3: The weights from solely the first network layer are transferred and those are not updated during retraining.



(OH). (OH)(N). (OH)(F-OH).

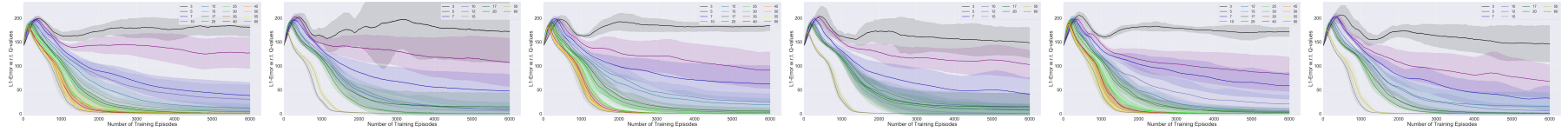
(f) Gridworld^{RT} 3x3: The weights from both network layers are transferred and updated during retraining.

Figure A.51: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the Gridworld^T 3x3 and Gridworld^{RT} 3x3 domains for each of the three forms of transfer. The first and second figure always show the retraining performance when the DQN has been trained on the original domain without auxiliary loss and with auxiliary loss, respectively. Values are based on at most 10 repetitions, because repetitions for which training a DQN with a hidden layer size of 3 leads to exploding gradients on the original domain when the auxiliary loss is employed are discarded. 95%-confidence intervals are shown.



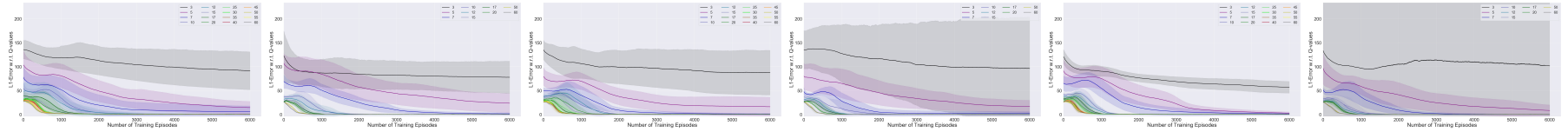
(OH). (OH)(N). (OH)(F-OH).

(a) FrozenLake^R 4x4: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



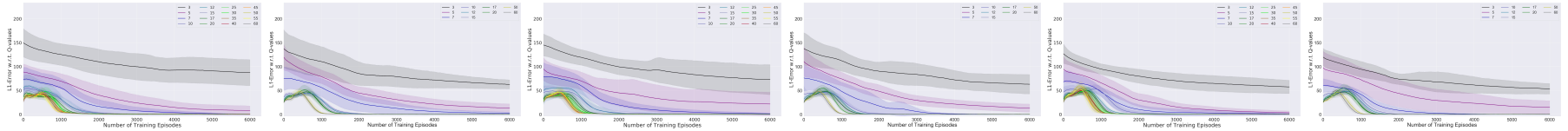
(OH). (OH)(N). (OH)(F-OH).

(b) FrozenLake^R 4x4: The weights from solely the first network layer are transferred and those are not updated during retraining.



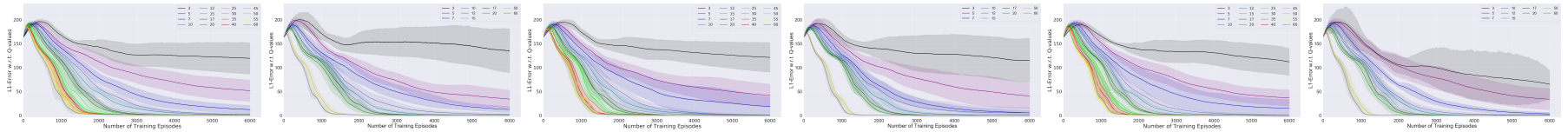
(OH). (OH)(N). (OH)(F-OH).

(c) FrozenLake^R 4x4: The weights from both network layers are transferred and updated during retraining.



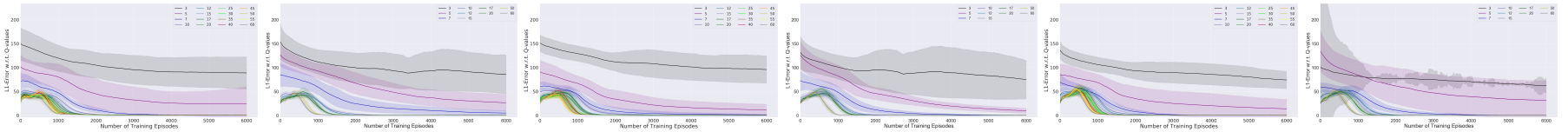
(OH). (OH)(N). (OH)(F-OH).

(d) FrozenLake^{RT} 4x4: The weights from both network layers are transferred, but solely those from the last layer are updated during retraining.



(OH). (OH)(N). (OH)(F-OH).

(e) FrozenLake^{RT} 4x4: The weights from solely the first network layer are transferred and those are not updated during retraining.



(OH). (OH)(N). (OH)(F-OH).

(f) FrozenLake^{RT} 4x4: The weights from both network layers are transferred and updated during retraining.

Figure A.52: Mean L_1-error with respect to the true Q-values during retraining of 2-layer DQNs with different hidden layer sizes for the FrozenLake^R 4x4 and FrozenLake^{RT} 4x4 domains for each of the three forms of transfer. The first and second figure always show the retraining performance when the DQN has been trained on the original domain without auxiliary loss and with auxiliary loss, respectively. Values are based on at most 10 repetitions, because repetitions for which training a DQN with a hidden layer size of 3 leads to exploding gradients on the original domain when the auxiliary loss is employed are discarded. 95%-confidence intervals are shown.

A.3.2. Robustness to Superfluous Feature Values Unseen During Training

Section A.3.2.1 provides methodological details and Section A.3.2.2 contains further figures.

A.3.2.1 Methodological Details

Motivation. Naturally, one would expect hidden-layer representations that do not distinguish states based on their superfluous feature value to generalize better to the values sampled at test time, and hence to return the correct optimal action for more generated states. To evaluate the usefulness of learning a hidden-layer representation that is similar to the coarsest Markov state representation, however, we need to train networks that differ in how close to the coarsest Markov state representation their formed hidden-layer representations are, while controlling for other factors that could impact the generalization behavior. Thereby, there are several options for creating such differing internal state representations based on our experimental results from Chapter 3:

- We could train networks with *different types of state encoding*, because our experimental results reveal that the hidden-layer representations learned differ between the types of state encoding, especially for larger-than-necessary hidden layer sizes. However, our types of state encoding also differ in the scale of the superfluous feature used during training, which makes a direct comparison more difficult[11].

- Another approach to generating such internal state representations is to train networks with *different hidden layer sizes*. Recall that our experimental results suggest that the extent to which the created first-layer state representation is similar to the coarsest Markov state representation depends on the hidden layer size, amongst others. Yet, this approach is taking advantage of the fact that neural networks with much larger-than-necessary capacities tend to overfit to the training data if no form of regularization is applied [18]. Thus, while the formed first-layer representation certainly is less close to the coarsest Markov state representation for very large hidden layer sizes, it also simply gets closer to the one in the input layer for much larger-than-necessary hidden layer sizes.

Table A.3: Hyperparameter values for the bisimulation-based auxiliary loss that is added to the training of 2-layer DQNs for the Gridworld 3x3 domain to explore the robustness to superfluous feature values unseen during training. Values depend on the hidden layer size (HLS) of a DQN.

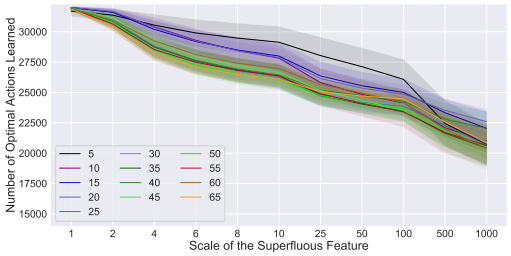
Parameter	Description	HLS 5	HLS > 5
d_E^{max}	Target Euclidean distance of the activations of two states with the max. bisimulation-based distance.	$\sqrt{32 \times HLS}$	$\sqrt{128 \times HLS}$
Decay rate	Decay rate, applied after each application of the auxiliary loss.	0.9999	0.9999
Measure	Bisimulation-based measure used for the auxiliary loss.	d_{fix}	d_{fix}
Stop episode	Training episode starting at which the auxiliary loss is no longer applied.	500	1,000
Weight	Weight of the auxiliary loss.	0.1	0.2

[11] For example, in the (OH) encoding, the scale of the superfluous feature is four times as large as the one of the features encoding the ground state, whereas the scales of the superfluous feature and of the features encoding the ground state are equivalent for the (OH)(N) encoding. Consequently, the difference in scales for the (OH) encoding may mean that for hidden layers that are slightly larger than necessary, the internal state representation formed in the first layer can generalize better to superfluous feature values outside the training interval, even though states from the same ground state and with superfluous feature values seen during training are mapped to less similar activations than for the (OH)(N) encoding.

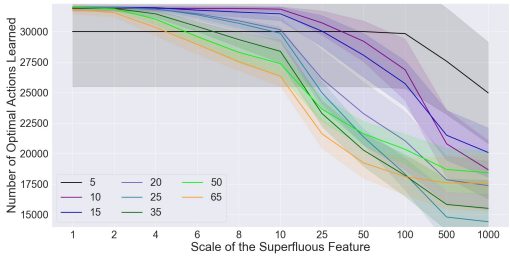
Auxiliary loss. Our chosen hyperparameter values for the auxiliary loss are listed in Table A.3. Thereby, the same two observations as in the context of transfer to related domains hold, which are important to keep in mind when analyzing our results. First, just-right hidden layer sizes require a first-layer representation coarser than the coarsest Markov state representation to be learned for the network to be able to converge to the true Q-values. Therefore, the auxiliary loss is applied less intensively for such hidden layer sizes. Second, the first-layer representations tend to become slightly less Markov again towards the end of training for very large hidden layer sizes such as 60 due to our hyperparameter settings for the auxiliary loss.

A.3.2.2 Further Figures

Figure A.53 depicts the average numbers of optimal actions learned by 2-layer DQNs with and without adding the bisimulation-based auxiliary loss to the training process for the Gridworld 3x3 (OH) and Gridworld 3x3 (OH)(N) domains.

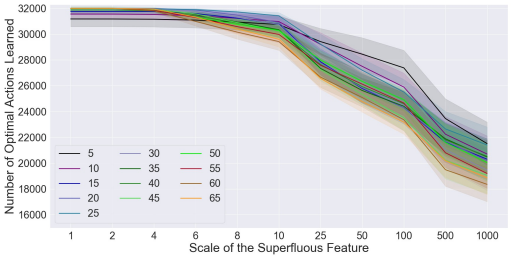


Without auxiliary loss.

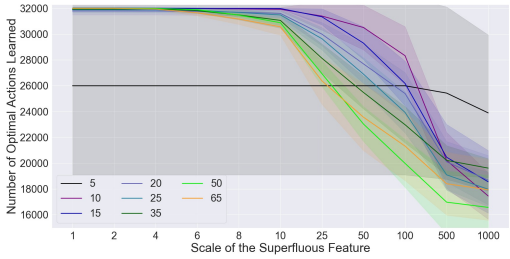


With auxiliary loss.

(a) Gridworld 3x3 (OH).



Without auxiliary loss.



With auxiliary loss.

(b) Gridworld 3x3 (OH)(N).

Figure A.53: Average numbers of optimal actions learned by 2-layer DQNs with different hidden layer sizes, with and without adding the bisimulation-based auxiliary loss to the training process. Optimal action returned for each non-terminal ground state are measured when sampling $1,000$ different values for the superfluous feature uniformly at random from an interval that is i times as large as the one used during training. The value i is depicted on the x-axis. Since there are 32 non-terminal states in the Gridworld 3x3 domain, returning $32,000$ optimal actions is optimal. Averages are based on 10 and 30 repetitions for the experiments with and without auxiliary loss, respectively. 95%-confidence intervals are shown.

B

Characteristics of Internal State Representations in Partially Observable Domains

In this section, we investigate the internal state representations formed by deep RL agents in partially observable domains. To this end, we explore the internal state representations present in the layers of DRQNs during and at the end of training in Section B.2. Our experimental approach is described in further detail in Section B.1.

B.1. Methodology

In the following, we describe in detail the computed correlation coefficients, domains, state encoding, network architectures and training procedure, and the t-SNE visualizations that we create for the internal state representations.

B.1.1. Correlation Coefficients

To compute Pearson correlation coefficients based on (components of) bisimulation metrics for our partially observable domains, we use the belief states corresponding to finite-length histories as states. Recall that since a belief state is a sufficient statistic for the agent's history of interactions with the environment and its initial belief state, a POMDP can be reduced to a continuous MDP in which the fully observable states are the internal belief states (see Section 2.3). Based on belief states, we calculate the following correlation coefficients:

1. $c_{d'_{fix}}$.

   - *Definition*. This is the Pearson correlation coefficient between the approximate Kantorovich distance-based bisimulation distances d'_{fix} of the belief states on the one hand and the Euclidean distances between the activations the belief states are mapped to in a neural network layer on the other hand.
   - *Computation*. We employ the approximation algorithm by [8] as described in Section 2.7.1.2 to compute d'_{fix}. Our chosen hyperparameters for this algorithm are listed in Section D.1.1 in the Appendix and the equation for subsequently computing the Pearson correlation coefficient is given in Equation 3.1.

2. c_{Rew}.

   - *Definition*. This is the Pearson correlation coefficient between the maximum absolute reward distances of the belief states on the one hand and the Euclidean distances between the activations the belief states are mapped to in a neural network layer on the other hand.

- *Computation*. Analogously to the computation for finite MDPs described in Section 3.1.1, the maximum absolute reward distance between two belief states b and b' is given by:

$$max_{a \in A} |R(b, a) - R(b', a)|. \tag{B.1}$$

  After the computation of the pairwise maximum absolute reward distances of belief states and the pairwise Euclidean distances between the activations belief states are mapped to, the Pearson correlation coefficient is computed (see Equation 3.1).

3. c_{TV}.
   - *Definition*. This is the Pearson correlation coefficient between the maximum total-variation distances T_{TV} of the transition functions of the belief states on the one hand and the Euclidean distances between the activations the belief states are mapped to in a neural network layer on the other hand.
   - *Computation*. The computation of c_{TV} proceeds analogously to the case of finite MDPs discussed in Section 3.1.1. Notice that when we compute the total-variation distance between belief states b_i and b_j for some action a, we do not enumerate all belief states as next states, because this would be infeasible. Instead, we only take the belief states $b_i^{a,o}$ and $b_j^{a,o}$, $o \in O$ s.t. $P(o|b_i, a) > 0$ or $P(o|b_j, a) > 0$, that result from making observation o after taking action a in belief states b_i and b_j, respectively, into consideration. Nevertheless, calculating c_{TV} for POMDPs is significantly more computationally expensive than for MDPs due to the computation of next belief states.

Since it is infeasible to calculate these correlation coefficients based on all possible belief states, we by default utilize a sample of 500 belief states based on histories that are collected by letting an agent walk randomly through the domain. Yet, some additional experiments are conducted to explore the impact of using a larger sample of belief states in Section C.7. Finally, since computing c_K proved not to be useful for finite MDPs, we do not calculate this correlation coefficient for POMDPs[1].

B.1.2. Domains

Experiments are conducted based on the Hallway domain [38][2], which consists of 15 grid locations as depicted in Figure B.1. In each grid position but the goal state, 4 orientations are possible, which leads to a total of 57 states. The start state is chosen uniformly at random from the non-terminal states and an episode ends when the goal state has been reached or when 100 steps have been executed. In the original implementation, the agent can choose from the actions $\{stay,\ forward,\ turn\ right,\ turn\ left,\ turn\ around\}$. Yet, we simplify the domain by removing the $turn\ around$ and $stay$ actions. A reward of 1 is obtained for arriving in the goal state and a reward of 0 for arriving in any other state. In each state, the agent generally observes the presence of a wall in each of the 4 possible locations. When the agent arrives in the goal state, however, it receives a distinct observation. In addition, when the agent faces south in one of the non-terminal states in the second row of the domain, a distinct observation is obtained. Lastly, while the transition probabilities are stochastic in the original description of the domain, deterministic transitions are utilized in this work. The reason for this choice is that the only computationally feasible algorithm for approximating d_{fix} for continuous MDPs is suitable solely for deterministic MDPs as described in Section 2.7.1.2.

B.1.3. State Encoding

Since we do not add a superfluous feature to the observations in the partially observable domain, the only question is how to encode the observation histories themselves. Evidently, one-hot encoding histories to ensure a Pearson correlation coefficient of 0 between the Euclidean distances of the encoded

[1] Moreover, it is not clear how to compute c_K based on belief states, since such a computation requires assigning identifying numbers to all belief states. It was possible to assign distinct numbers to each state in our small fully observable domains. Yet, assigning unique numbers to each possible belief state to subsequently compute the cumulative distribution function is infeasible. While one could consider only the next belief states with non-zero probabilities of being reached when computing T_K for two histories, such computations of T_K would not always assign the same identifiers to a next belief state. Thus, two next belief states could be assigned very similar identifiers during one and more different ones during another computation of T_K. Calculating c_K based on such pairwise distances of histories would therefore be even less meaningful than in the context of MDPs.

[2] We implemented the Hallway domain in the pomdp_domains package.

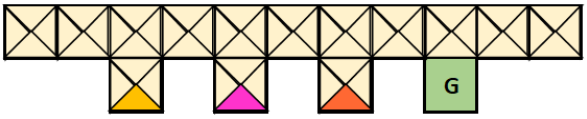


Figure B.1: Hallway domain. The triangles denote the possible orientations of the agent and the brightly colored triangles mark states in which distinct observations are received by the agent. The goal state is specified by G.

histories and the bisimulation-based measures is not feasible. Therefore, we opt to one-hot encode the observations. This not only allows for a reasonable dimensionality of the encoded observations but also guarantees that all input dimensions have the same scale.

B.1.4. DRQN Implementation and Training

DRQNs are implemented in PyTorch 1.3.1 and trained as described in Section 2.4.1 with the Adam algorithm. Thereby, a target network and a replay memory are used to improve learning stability as outlined in Section 2.4.1. Each trained DRQN consists of three layers, of which the first and the last one are linear layers and the second one is an LSTM layer. Each hidden layer has the same size, which is given by the $layer\_size$ parameter in Table B.1, unless an experiment involves changing the hidden layer size. In addition, *Rectified Linear Unit* (ReLU) activations are utilized for each hidden layer. The policy of the agent during training is ϵ-greedy. In case ϵ is decayed during the training process, ϵ is multiplied by the $decay\_rate$ parameter after each episode whose terminal reward is higher than the current average of terminal rewards. The hyperparameter values are determined via grid search and the final values are listed in Table B.1.

Table B.1: Hyperparameter values for DRQNs for the Hallway domain.

Parameter	Description	Value
batch_size	Batch size for training	128
decay	Whether to decay ϵ	True
decay_rate	Decay rate for ϵ	0.999995
discount_factor	Discount factor of POMDP	0.95
history_length	Length of histories	7
layer_size	Size of hidden layers	32
learning_rate	Learning rate α	0.00005
lr_decay	Whether to decay α	True
lr_decay_rate	Decay rate for α	0.99995
max_epsilon	Starting value for ϵ	1
max_steps	Max. # of steps per episode	100
min_epsilon	Min. value for ϵ	0.35
num_episodes	Number of training episodes	30,000
replay_memory_size	Capacity of replay memory	50,000
target_update	Update frequency for target network	150
weight_ep	Sampling weight for successful episodes (1 is normal)	1
weight_hist	Sampling weight for histories ending in the goal state (1 is normal)	10

B.1.5. Convergence of Test Rewards

To determine when an agent has learned to always reach the goal state, we test each DRQN 100 times after each training episode.

B.1.6. t-SNE Visualization

To visualize the representations learned by a DRQN, t-SNE plots of the activations in a network layer are created as explained in Section 2.8 with a default perplexity value of 30. Thereby, either all the

activations of histories belonging to belief states with the same most probable ground state are drawn in the same color as much as possible, or activations are colored based on clustering histories with respect to their bisimulation-based distances. These two coloring types are discussed below.

B.1.6.1 Coloring Based on the Ground State

The activations of each history are colored based on the ground state the agent believes to be most likely after acting according to that history. If there are multiple ground states with the highest belief value for a history, the activations for that history are colored based on the ground state with highest belief value that has the lowest index and with a pink border. Indices are assigned to ground states from left to right based on the underlying grid as shown in Figure B.2. Regarding the ground state colors, each ground state is assigned a unique color and ground states with the same actions corresponding to shortest paths to the goal state receive similar colors. For example, all ground states for which the action $turn\ right$ leads to the goal state most quickly are drawn in shades of blue and purple. Notice, however, that these actions corresponding to the fastest path to the goal are not necessarily the optimal actions, since action my also be taken to gather information. In addition, the goal state is depicted in pink. The detailed coloring scheme for the ground states is shown in Figure B.3.

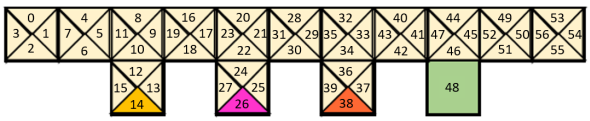


Figure B.2: Ground state indices for the Hallway domain.

Figure B.3: Colors used for each ground state of the Hallway domain in t-SNE plots. Each ground state is assigned a unique color and ground states with the same actions corresponding to shortest paths to the goal state receive similar colors.

B.1.6.2 Coloring Based on K-Means Clustering

Coloring activations based on the ground state corresponding to the highest belief value leads to a loss of information when there are multiple such ground states. Therefore, we also color activations in t-SNE plots by clustering the activations that histories are mapped to based on a matrix with pairwise distances based on d'_{fix} via the K-Means algorithm implemented in the Scikit-learn package. Thereby, we create 5, 10, 15 or 20 clusters.

B.2. Results

We delineate the learning process for DRQNs in Section B.2.1 and describe factors impacting the hidden-layer state representations in Section B.2.2.

B.2.1. Learning Process

Recall that we identified three overlapping learning phases for DQNs in Chapter 3. Our experimental results for DRQNs suggest that similar learning phases also occur for partially observable domains, as we will outline subsequently.

B.2.1.1 Learning Phase 1

At the beginning of training, the internal state representations tend to cluster the activations of belief states based on multi-step rewards, depending on the number of times the target network has been updated. Yet, histories are not fully clustered based on immediate rewards until after the target network has been updated once. Since the reward-based correlation coefficient c_{Rew} measures the degree to which belief states are grouped based on immediate rewards, this can be seen from a strong increase in c_{Rew} once the target network has been updated once and a subsequent step-wise decrease in c_{Rew} (see Figure B.7).

Calculating bisimulation-based correlation coefficients thus is very promising to gain insights into the internal state representations also of DRQNs. t-SNE plots of the activations states are mapped to, however, are much less helpful for POMDPs than for the small MDPs discussed in Chapter 3. For instance, one can see that the peak value for c_{Rew} is accompanied by a distinct cluster of states at the bottom of t-SNE plot 2 in the first row of Figure B.4, which visualizes the output-layer representations of a DRQN during training. Yet, it is not clear which belief states correspond to this cluster of activations. This is the case, because the cluster includes several activations of belief states for which multiple ground states are believed to be most likely[3], which means that the coloring based on ground states does not allow to uniquely identify the belief states.

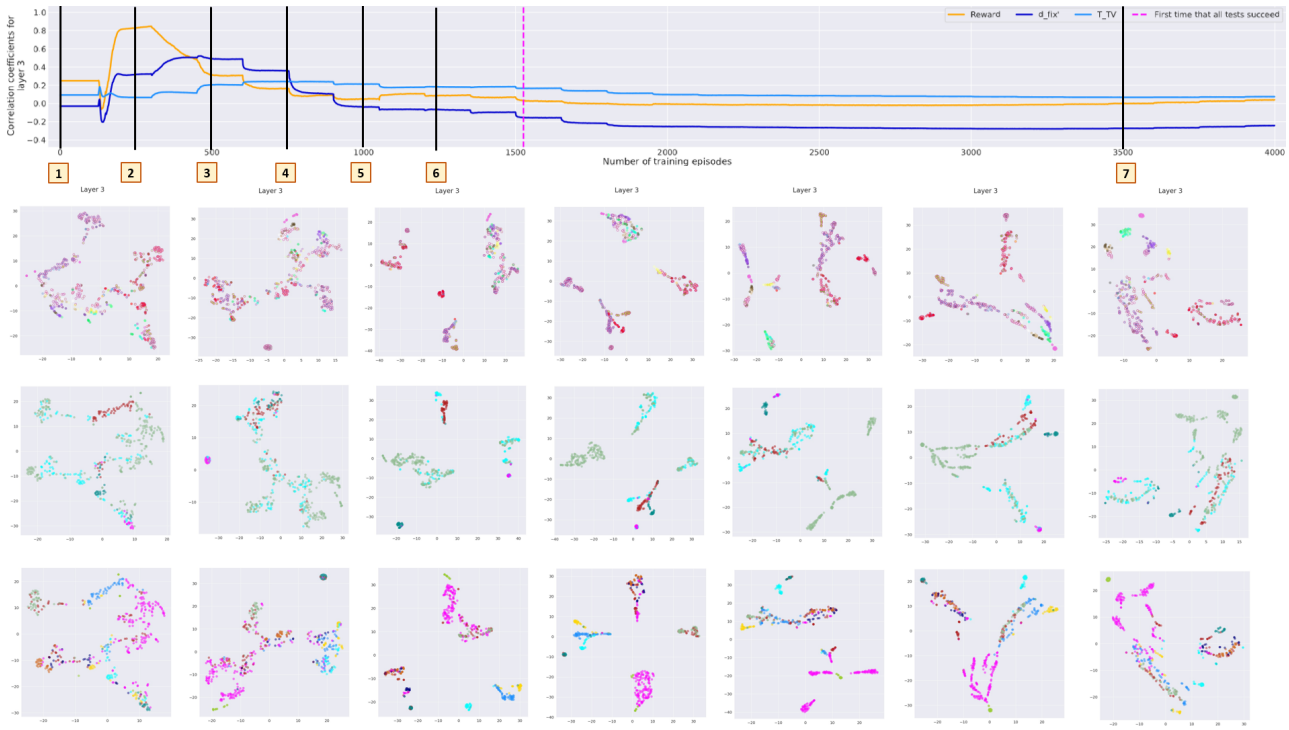


Figure B.4: Correlation coefficients and t-SNE plots of the activations histories are mapped to during training for the output layer of a DRQN for the Hallway domain. The hidden layer size is equal to 16. In the first row, activations are colored based on the corresponding most likely ground states and in the second and third row based on 5 and 10 clusters based on d'_{fix}, respectively.

B.2.1.2 Learning Phase 2

As the network layers' internal state representations group belief states based on multi-step rewards, the representations become more similar to the coarsest Markov state representation. This is visualized in Figure B.7, where $c_{d'_{fix}}$ increases at the beginning of training in all three layers in a way that is similar to the one observed for MDPs in Chapter 3. Again, looking at the t-SNE plots in Figure B.4 in which the activations are colored based on the ground states does not provide many insights as to which belief states are now grouped together. This is the case, because there is a large number of belief states with multiple equally probable ground states. However, the t-SNE plots in which activations are colored based on clusters created via the K-Means algorithm clearly show that a high value for $c_{d'_{fix}}$ corresponds to an internal state representation in which belief states that have a low distance with respect to d'_{fix} tend to be mapped to activations that are relatively close in Euclidean distance.

We further make the following observations with respect to the second learning phase:

- *The first-layer representation is more similar to the coarsest Markov state representation than the ones in the LSTM and the output layer throughout training.* The first-layer representation has the highest value for $c_{d'_{fix}}$ throughout training as depicted in Figure B.7. Yet, notice that the correlation coefficients for the first layer only measure where histories with the same last

[3]Such activations have a pink border in the t-SNE plots.

observation are mapped. This is the case, because there is no hidden state that impacts the first-layer activations a history is mapped to and because we compute the correlation coefficients based on the activations of solely the last observations of histories. Therefore, the observation that the first layer has the highest value for $c_{d'_{fix}}$ throughout training is partially due to the fact that histories with low bisimulation-based distances also tend to have similar last observations for this domain, as mirrored by the high initial first-layer value for $c_{d'_{fix}}$ of around 0.4 (see Figure B.7).

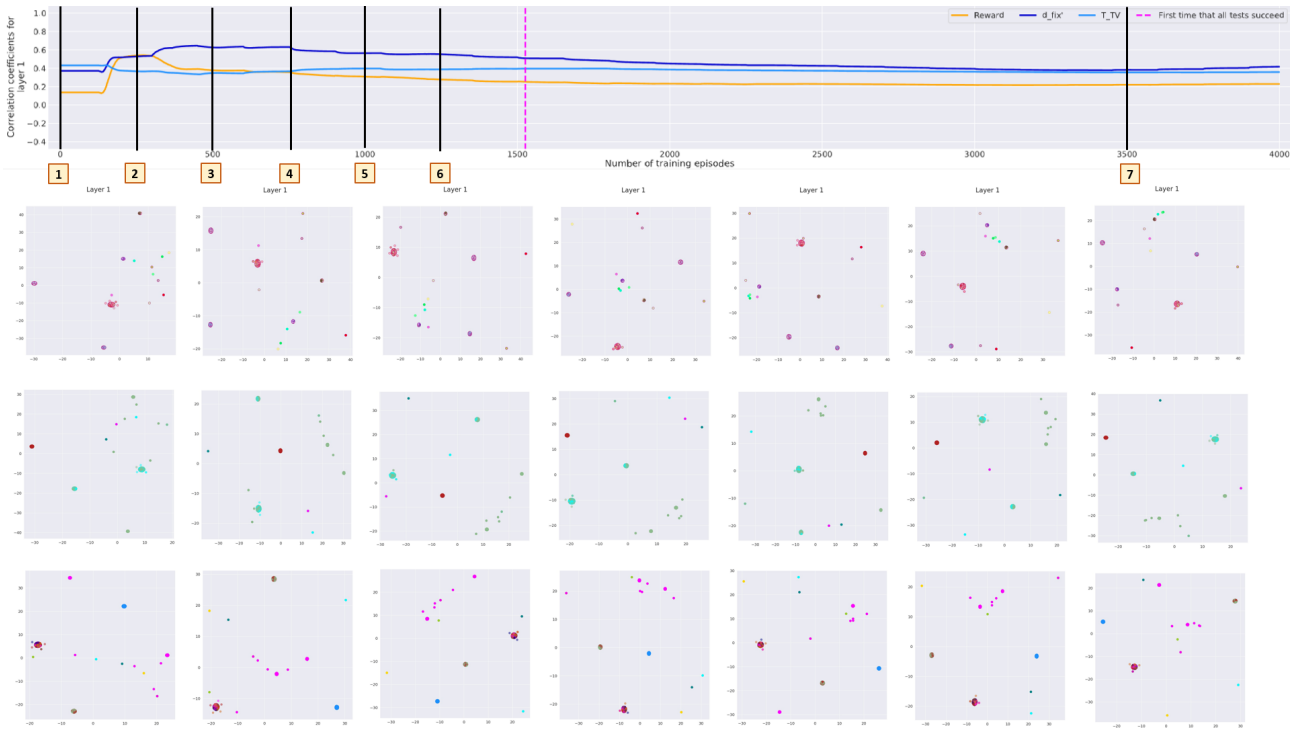


Figure B.5: Correlation coefficients and t-SNE plots of the activations histories are mapped to during training for the first layer of a DRQN for the Hallway domain. The hidden layer size is equal to 16. In the first row, activations are colored based on the corresponding most likely ground states and in the second and third row based on 5 and 10 clusters based on d'_{fix}, respectively.

Nevertheless, $c_{d'_{fix}}$ also increases during training for the first layer. Figure B.5 shows that there are initially 14 clusters in the first layer, one for each observation that can be made in the domain[4]. The first-layer representation then becomes more similar to the coarsest Markov state representation by mapping different observations with relatively low pairwise distances based on d'_{fix} to more similar activations. This is visible in the t-SNE plots in Figure B.5 that color activations based on clustering via K-Means. For instance, the distinct observations that are made when facing south in one of the non-terminal ground states in the second row of the domain are mapped to more similar activations.

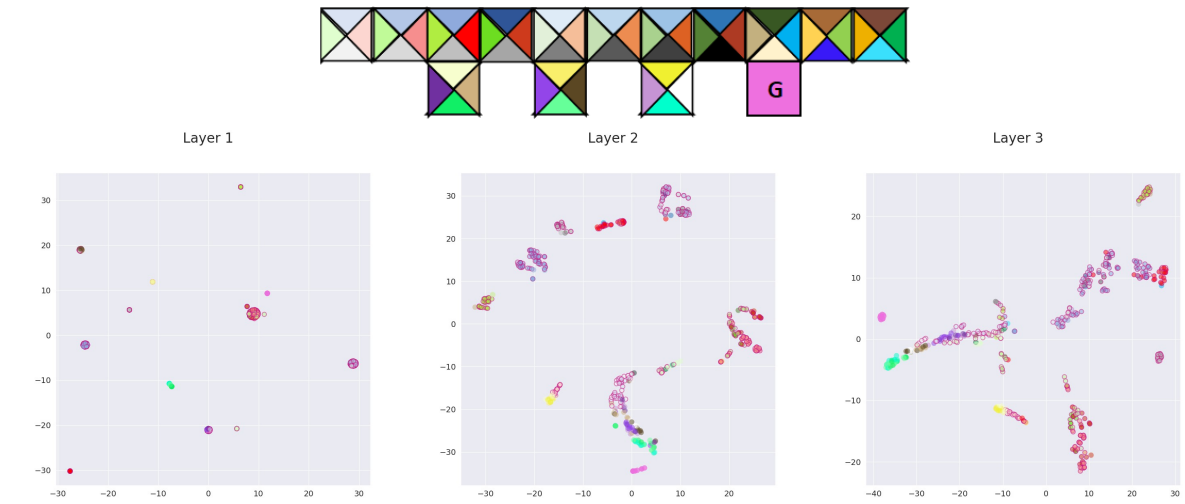
- *We cannot conclude that the LSTM layer forms an internal state representation that is more similar to the coarsest Markov state representation during the second learning phase than the one in the output layer.* In the context of MDPs, our results indicate that the first-layer representation generally is more similar to the coarsest Markov state representation than both the input- and the output-layer representation during the second learning phase. Since the first layers of our DRQNs consider merely single observations for their internal state representations, one might therefore assume that the LSTM layer creates an internal state representation that is more similar to the coarsest Markov state representation than the one in the output layer during this learning phase. Yet, we find that while the LSTM-layer representation tends to be more Markov than the output-layer one at the end of training, this is not the case for the beginning of training while histories are increasingly clustered based on multi-step rewards.

[4] If we used the original stochastic observation function of the Hallway domain, the observation space size would be 20. Not all of these observations are possible when the presence of walls is always observed correctly, however.

That it is not observed in our results that the LSTM-layer representation becomes more similar to the coarsest Markov state representation than the output-layer representation at the beginning of training could be due to two different aspects. First, the smallest hidden layer size for which we performed experiments is 8, but we do not know whether this is the smallest hidden layer size for which training is successful for the Hallway domain. It could thus be that for even smaller network capacities, the LSTM-layer representation becomes more Markov than the one in the output layer at the beginning of training. Second, our experiments with MDPs suggest that for domains with larger state spaces, the first-layer representation becomes more Markov than the one in the output layer to a lesser degree than for domains with smaller state spaces. Note that the number of belief states for the Hallway domain is higher than the number of states in any of our fully observable domains and that we do not know whether a hidden layer size of 8 is larger than necessary. It could consequently be the case that it takes longer to form a representation that is Markov based on a given number of transitions in the LSTM layer, and that the output layer hence learns to represent rewards based on a certain number of steps before the LSTM layer has fully formed such a representation.

B.2.1.3 Learning Phase 3

Our experiments with DQNs in Chapter 3 clearly show that the output layer ultimately forms a Q^*-irrelevance abstraction and that the hidden layers also progressively cluster states based on Q-values at some point during training. Thereby, we could draw our conclusions based on the value of the correlation coefficient c_{Q^*} and t-SNE plots in addition to the fact that DQNs are trained to predict Q-values. Because we do not calculate c_{Q^*} for our experiments with DRQNs, we have to rely on t-SNE plots and the knowledge that DRQNs, too, are trained to learn Q-values alone:



(a) First layer. (b) LSTM layer. (c) Output layer.

Figure B.6: t-SNE plots of the activations histories are mapped to at the end of training for the layers of a DRQN for the Hallway domain. The hidden layer size is equal to 16. Activations are colored based on the corresponding most likely ground states. The coloring scheme for the ground states that are believed to be most probable is given at the top. If there are multiple most probable ground states for a belief state, the corresponding activation is depicted with a pink border.

- *Output layer.* The t-SNE plot for the output-layer representation at the end of training in Figure B.6c depicts five easily identifiable (sub-) clusters, which are those of activations drawn in shades of pink, purple, brown, yellow and blue-green. The cluster of activations drawn in pink thereby corresponds to belief states for which the terminal state is most likely, and the other four colors denote belief states with the highest belief for ground states near the distinct observations made when facing south in one of the non-terminal states in the second row of the domain. Since the activations for belief states for which the terminal state is most likely fall into a separate cluster, the activations colored in shades of purple, brown and blue-green are part of the same cluster but form distinct sub-clusters, and the activations drawn in shades of yellow form a distinct cluster, it seems likely that the DRQN has indeed converged to group states based on Q-values. Notice,

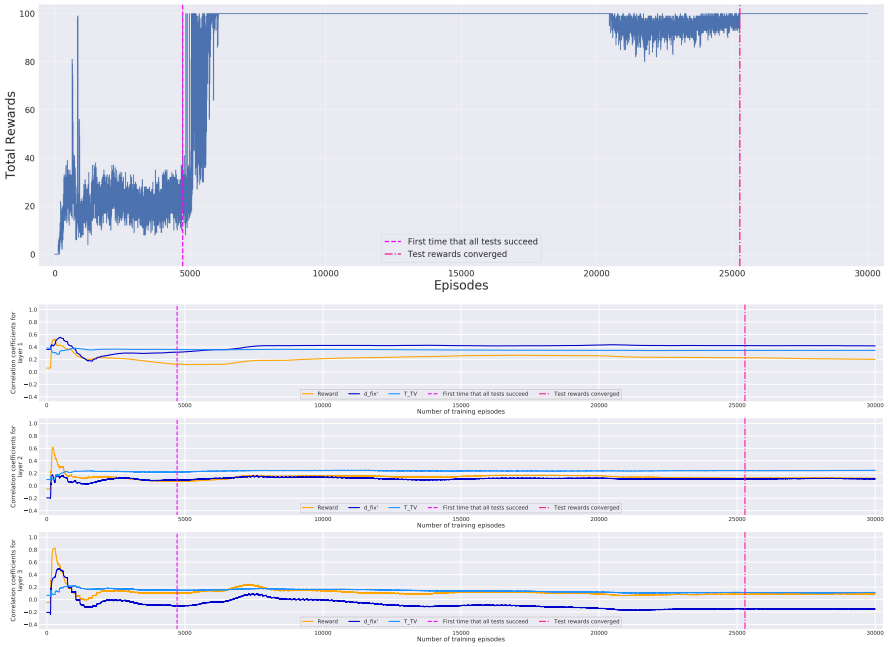
for instance, that the terminal ground state is the only ground state with Q-values of 0 for each action. Finally, the negative value for $c_{d'_{fix}}$ for the output layer at the end of training suggests that belief states for which the DRQN learns similar Q-values do not necessarily also have low distances with respect to d'_{fix} for this domain.

- *Hidden layers.* The fact that similar clusters of activations are present in the LSTM layer at the end of training as in the output layer suggests that the LSTM layer also progressively clusters belief states based on Q-values (see Figure B.6b). Moreover, $c_{d'_{fix}}$ decreases after an initial peak for the LSTM layer, just as it is observed for the layers of DQNs and for the output layer of DRQNs (see Figure B.7). The latter observation can also be made for the first layer, because Figure B.5 shows a decrease in $c_{d'_{fix}}$ for the first layer after an initial peak. This suggests that observations are also increasingly clustered based on Q-values in the first layer. Recall that we cannot expect to see a similar t-SNE plot of the activations at the end of training for the first layer as for the output layer, because the first layer solely clusters single observations rather than histories (see Figure B.6a).

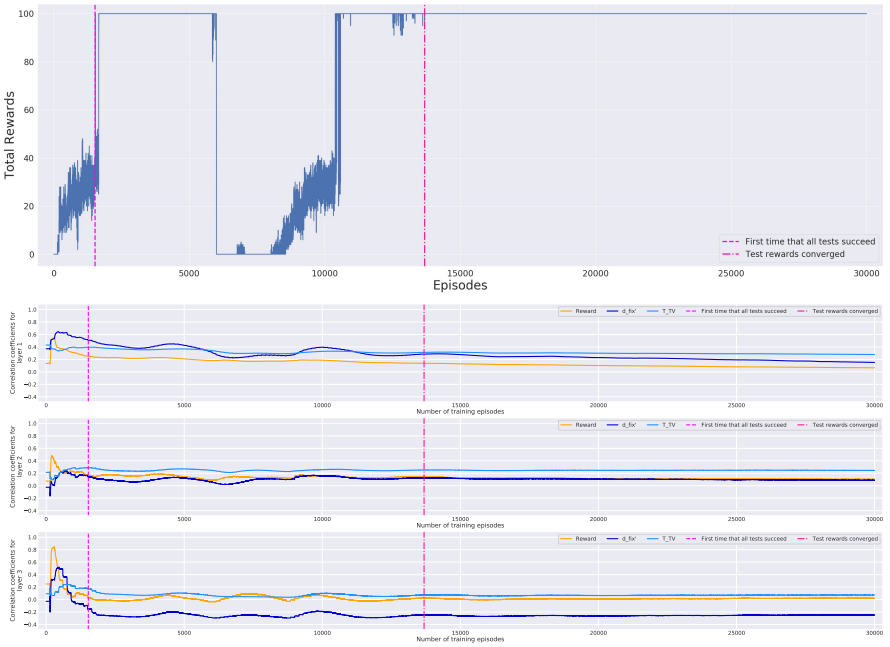
B.2.2. Factors Impacting the Hidden-layer State Representations

Our experimental results suggest that the extent to which the first- and the LSTM-layer representations become similar to the coarsest Markov state representation at the beginning of training and still are at the end of training depends on the hidden layer size. More precisely, Figure B.7 visualizes that $c_{d'_{fix}}$ is higher during the second learning phase for DRQNs with smaller hidden layer sizes. Recall that we also found this to be the case for MDPs in Section 3.2, and that this phenomenon can be explained by the fact that networks with higher capacities need less coarse of an abstraction of the state space to be formed in hidden layers to be able to learn the true Q-values than networks with lower capacities do.

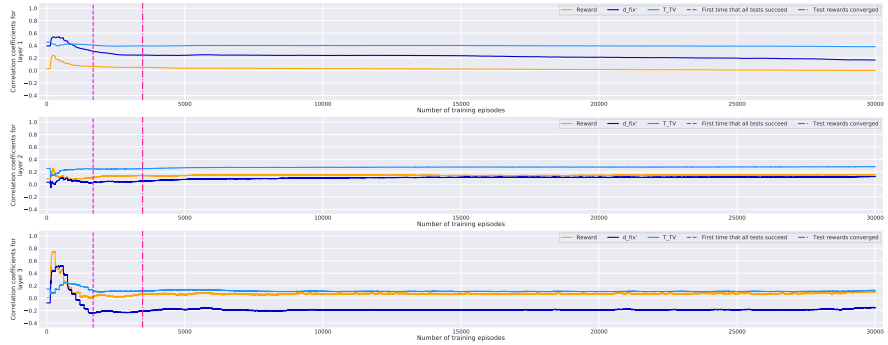
Furthermore, just as in the context of MDPs, the degree to which the internal state representations in hidden layers are similar to the coarsest Markov state representation at the end of training also depends on the hidden layer size. Specifically, the state representations in hidden layers are more similar to the coarsest Markov state representation at the end of training for smaller hidden layer sizes. This is largely due to the fact that the representations also become less similar to the coarsest Markov state representation and remain closer to the initial ones at the beginning of training for larger hidden layer sizes, because abstracting the state space is required to a lesser extent for networks with larger capacities. Yet, we cannot conclude based on our results that the hidden-layer representations of DRQNs with smaller hidden layers also tend to be abstracted towards towards a Q^*-irrelevance abstraction to a larger degree than those of DRQNs with slightly larger hidden layers, as we did in the context of MDPs. This is the case, because we only performed experiments with four different hidden layer sizes for DRQNs for the Hallway domain.



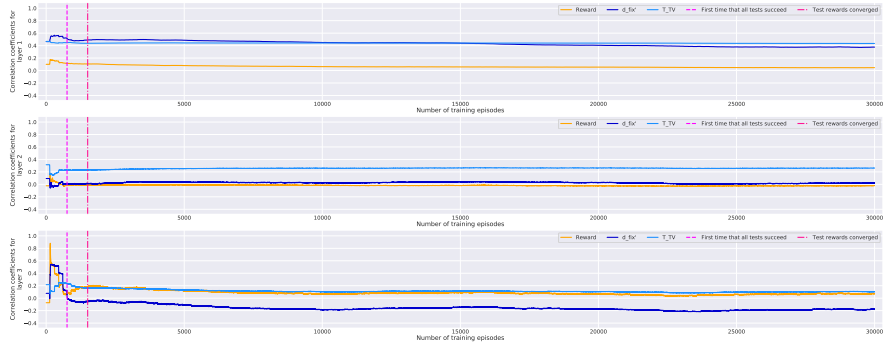
(a) $hidden\_layer\_size = 8$.



(b) $hidden\_layer\_size = 16$.



(c) $hidden\_layer\_size = 32$.



(d) $hidden\_layer\_size = 64$.

Figure B.7: Number of successful test episodes out of 100 for each training episode and c_{Rew}, $c_{d'_{fix}}$ and c_{TV} in each of the three layers of DRQNs for the Hallway domain during training for different hidden layer sizes. The first layer is a linear layer and the second layer an LSTM layer. The correlation coefficients are computed based on the belief states resulting from sampling 500 histories. The target network is updated every 150 training episodes.

C

Further Experiments

C.1. Fixed Replay Memory

The goal of the experiments in this section is to explore to which extent the patterns observed in Chapter 3 are merely due to the changing composition of the replay memory during training. To this end, 2-layer DQNs with a hidden layer size of 50 are trained for Gridworld 3x3 (OH) and FrozenLake 4x4 (OH). However, unlike in Chapter 3, the replay memory is not filled during training. Instead, a fixed replay memory is utilized for each domain, which is the one that exists at the end of training a 2-layer DQN without fixed replay memory as in Chapter 3. Since the replay memory size is larger than the total number of transitions performed during training, the replay memory contains all transitions encountered during training of a 2-layer network without fixed replay memory. The occurrences of the state-action combinations in the fixed replay memory used for Gridworld 3x3 (OH) are exemplarily shown in Figure C.1. While more transitions exist for optimal actions taken in non-terminal states than for non-optimal actions, especially when these non-terminal states are close to a goal state, each action is taken at least 7 times in each state. Notice that all DQNs that are trained with a fixed replay memory in this section still converge to the optimal policy and the true Q-values.

Figures C.2 and C.3 show for the FrozenLake 4x4 (OH) and the Gridworld 3x3 (OH) domain, respectively, that using a fixed replay memory does not significantly impact the final state representations learned in the network layers. For Gridworld 3x3 (OH) and the first layer for FrozenLake 4x4 (OH), employing a fixed replay memory also does not significantly change the way the final state representations are learned. For the second layer for FrozenLake 4x4 (OH), however, utilizing a fixed replay memory causes c_{TV}, c_K and c_{Rew} to increase more quickly at the beginning of training. This is due to the DQN learning to form clusters for ground states farther away from the start state more rapidly when a fixed replay memory is used. Notice that when no fixed replay memory is utilized, it takes some time for the agent to reach farther away states due to the holes that directly terminate an episode. Transitions containing such states are, however, already present in the replay memory at the beginning of training when a fixed replay memory is employed. This also explains why using a fixed replay memory has a much smaller impact on the way the second-layer state representation is learned for Gridworld 3x3 (OH). The reason is that this domain has neither holes that prematurely end an episode nor a fixed start state, which implies that a non-fixed replay memory has a much more even distribution over states at the beginning of training for this domain.

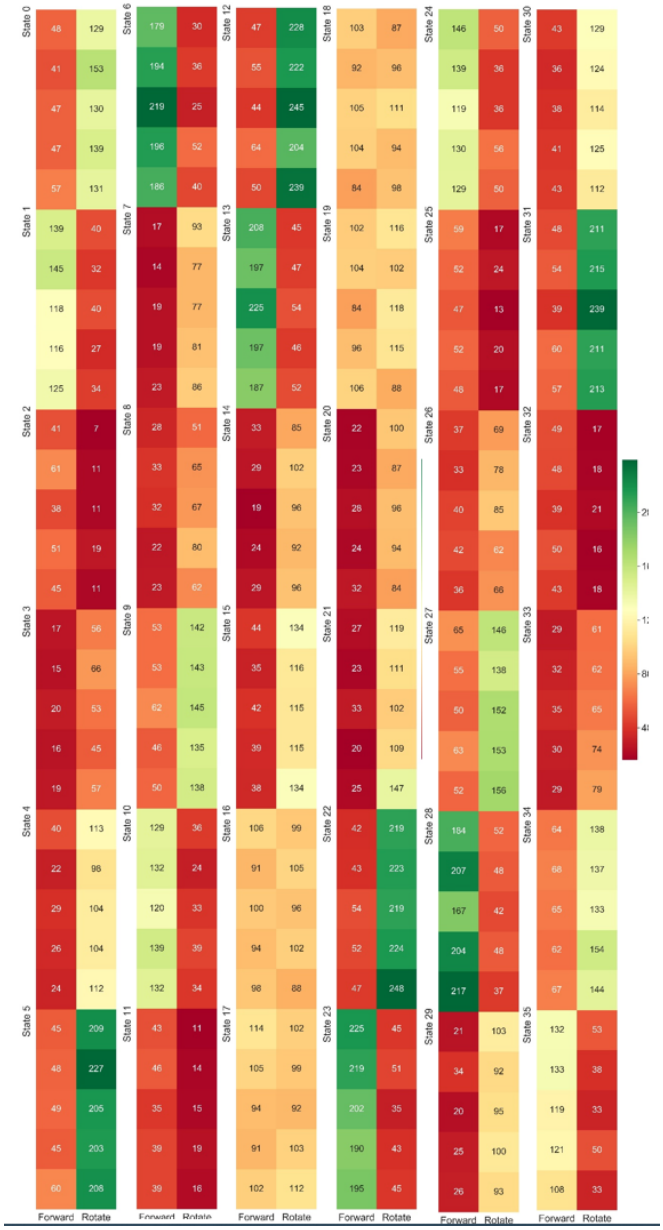
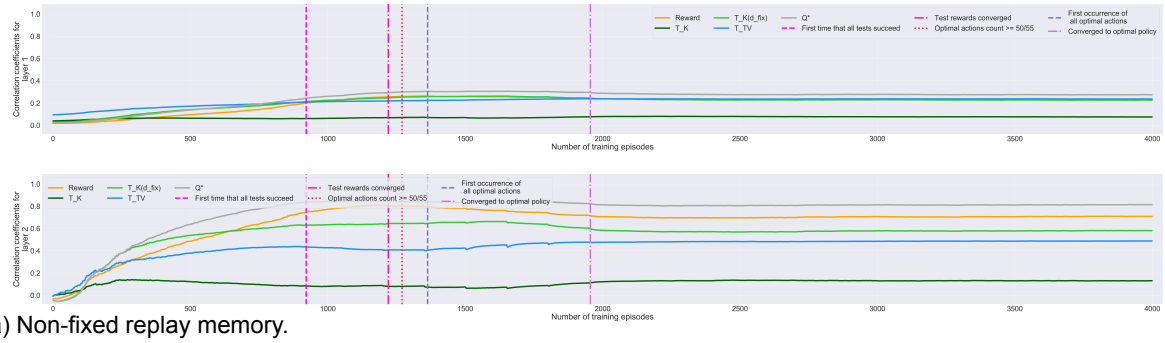
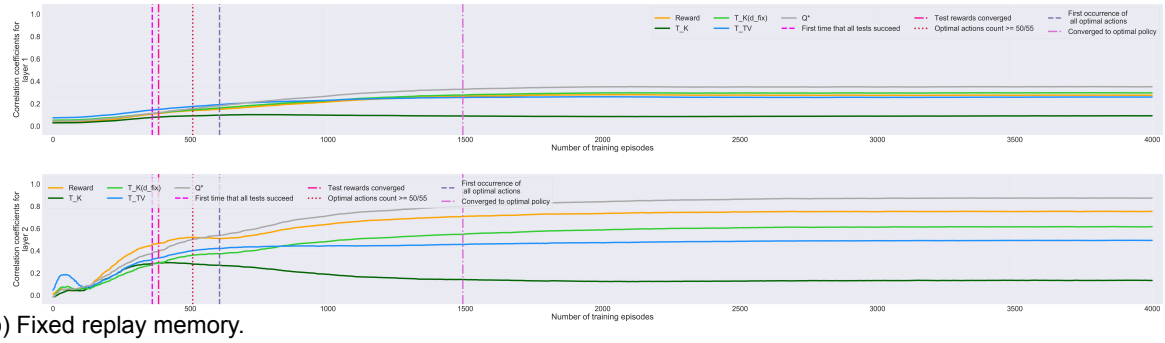


Figure C.1: Occurrences of state-action combinations in the fixed replay memory used for Gridworld 3x3 (OH). The y-axis shows the start state of a transition, grouped by ground state, and the x-axis the action taken in that state. All states belonging to ground states 16 through 19 are terminal states.

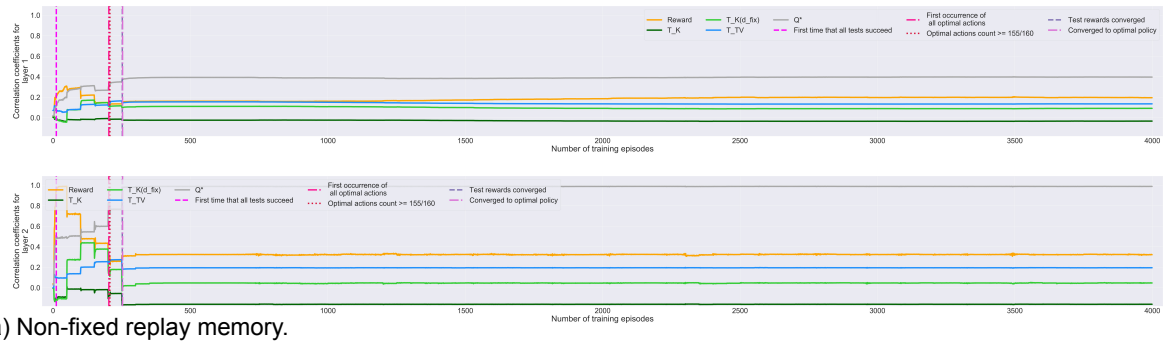


(a) Non-fixed replay memory.

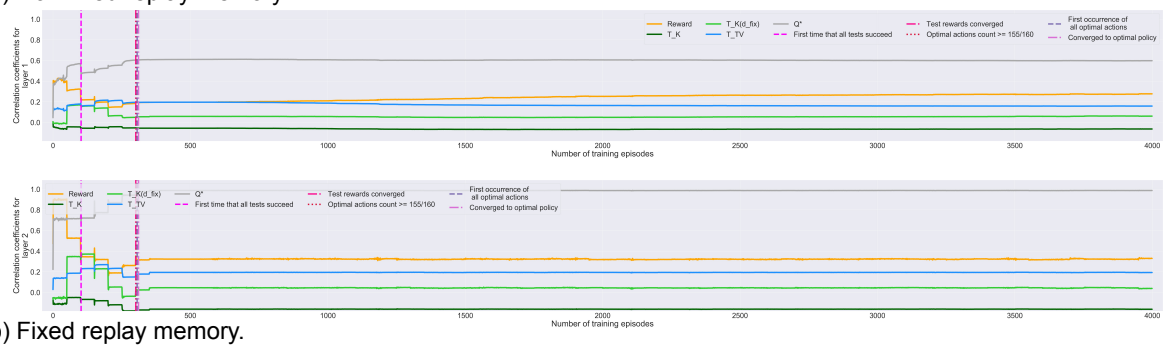


(b) Fixed replay memory.

Figure C.2: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for different numbers of training episodes for both layers of a 2-layer DQN for FrozenLake 4x4 (OH) when employing either a fixed or a non-fixed replay memory. The hidden layer size is equal to 50.



(a) Non-fixed replay memory.



(b) Fixed replay memory.

Figure C.3: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for different numbers of training episodes for both layers of a 2-layer DQN for Gridworld 3x3 (OH) when employing either a fixed or a non-fixed replay memory. The hidden layer size is equal to 50.

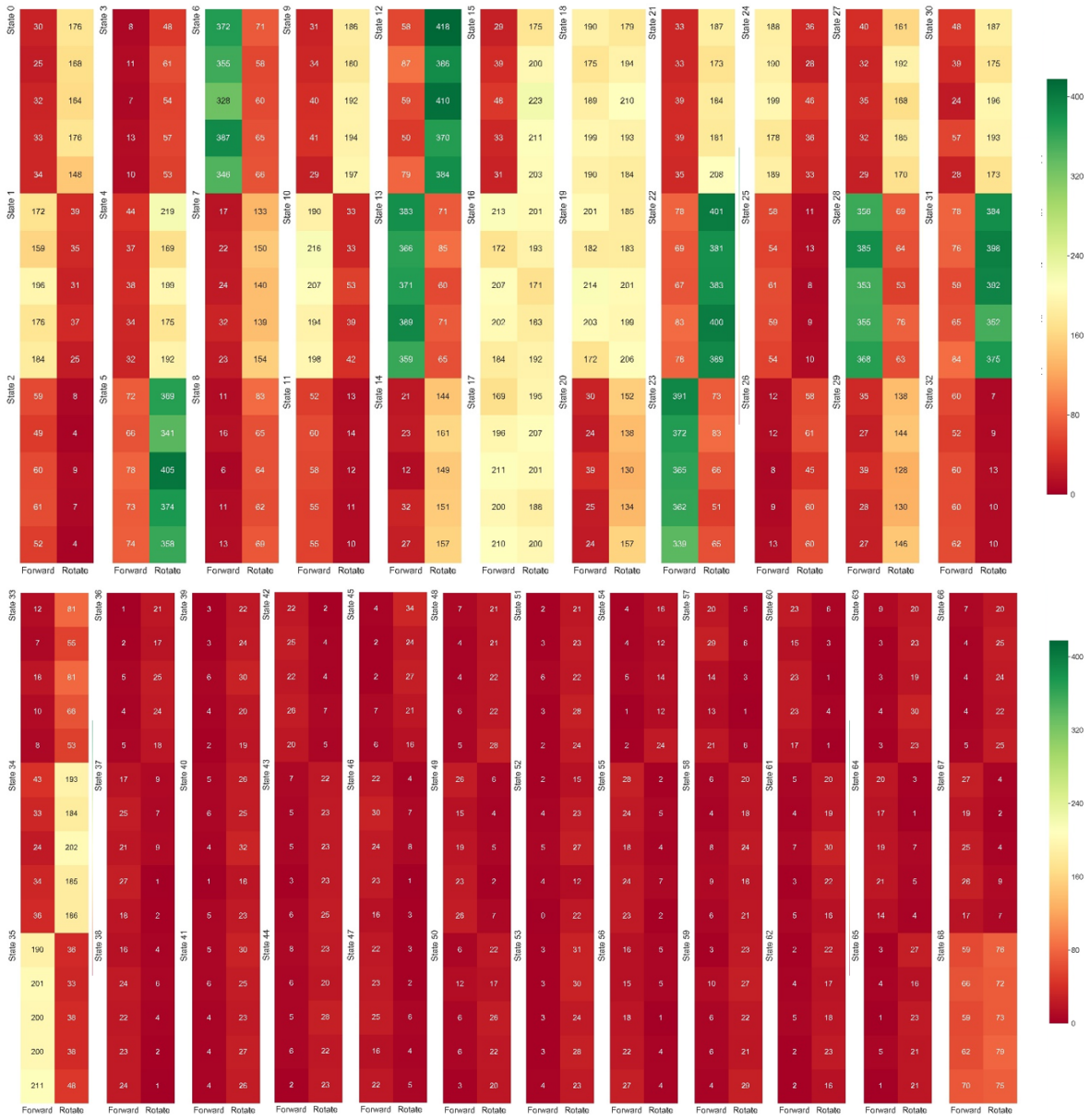
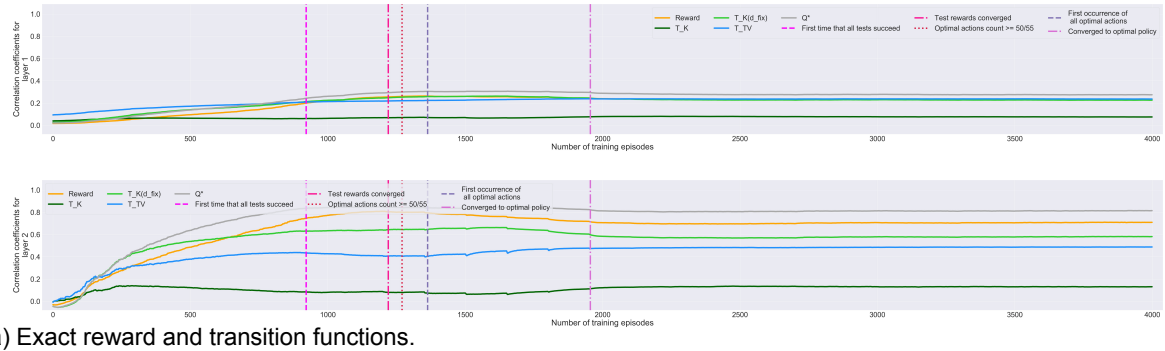


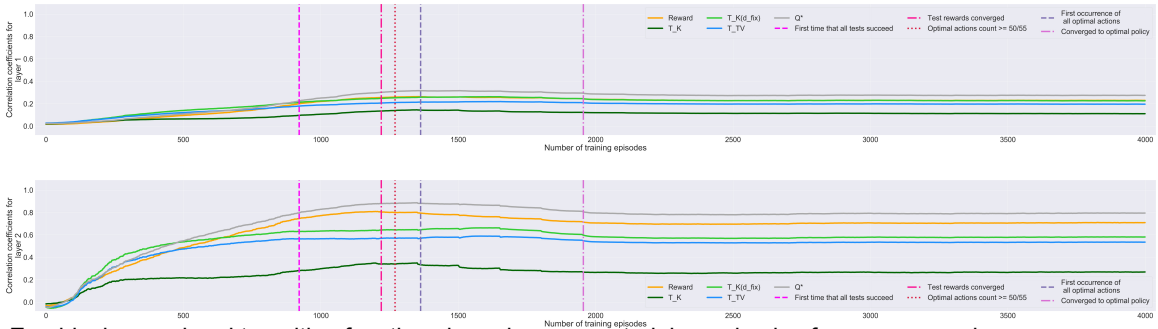
Figure C.4: Occurrences of state-action combinations in the fixed replay memory used for Gridworld 3x3 (Aug) (OH)(N). The y-axis shows the start state of a transition, grouped by ground state, and the x-axis the action taken in that state. All states belonging to ground states 16 through 19 as well as the last 5 states are terminal states.

C.2. Empirical vs. Exact Reward and Transition Functions to Compute c_K, c_{Rew} and c_{TV}

The experiments conducted in Chapter 3 compute bisimulation metrics and their components based on the exact reward and transition functions just like in the original work on bisimulation metrics [14]. Yet, utilizing the empirical reward and transition functions may be more generally applicable, because the true reward and transition functions of a domain may not be known in practice or the domain may be too large to store the true reward and transition functions exactly in a table.



(a) Exact reward and transition functions.



(b) Empirical reward and transition functions based on $4,000$ training episodes for c_{Rew}, c_K and c_{TV}.

Figure C.5: Correlation coefficients for different numbers of training episodes for both layers of a 2-layer DQN for FrozenLake 4x4 (OH). c_{Rew}, c_K and c_{TV} are calculated based on either the exact reward and transition functions or the empirical reward and transition functions that are present after $4,000$ training episodes. The other correlation coefficients are always computed based on the exact reward and transition functions. The hidden layer size is equal to 50.

Figure C.5 visualizes for a 2-layer DQN for the FrozenLake 4x4 (OH) domain that the observed values for c_K, c_{Rew} and c_{TV} do not significantly differ when using empirical rather than exact reward and transition functions to compute these three correlation coefficients. More precisely, the converged value of c_{Rew} in both layers does not change at all, and c_{TV} and c_K are slightly higher at the end of training when empirical transition and reward functions are used. The lack of change in the converged value of c_{Rew} can be explained by the fact that the tested domain is so small and simple that after the end of training, the empirical reward function is equal to the exact reward function. Furthermore, the empirical transition function only differs from the true transition function in that the probability of transitioning from one state s to another state s' is not exactly equal to 0.2 when the probability of transitioning from the ground state of state s to the ground state of state s' is equal to 1[1]. Thus, c_{TV} and c_K are higher when empirical transition functions form the basis of the computation, as the learned state representation is based on the transitions that the agent has actually seen rather than the exact transition probabilities.

Since FrozenLake 4x4 is a very small domain even with the added superfluous feature, experiments in larger domains are needed to explore whether using the empirical reward and transition functions rather than the exact ones is a viable option when the empirical reward and transition functions are less close to the exact ones. In addition, we did not compute $c_{K(d_{fix})}$ based on empirical transition and reward functions, which should also be investigated in the future.

[1]This exact value of 0.2 is due to uniformly at random choosing one of the five possible values for the added superfluous feature.

C.3. Approximately Computing d_{fix}

Computing $c_{K(d_{fix})}$ precisely is prohibitive for large domains, because the exact calculation of d_{fix} is expensive even for moderately sized domains. We thus subsequently compare using d_{fix} for measuring how close to the coarsest Markov state representation an internal state representation is to utilizing d'_{fix}. Recall that d'_{fix} is the result of approximating d_{fix} by means of the approximation algorithm by [8] delineated in Section 2.7.1.2. Notice that while computing d_{fix} with our selected precision takes several hours for FrozenLake 4x4, calculating d'_{fix} requires solely a few seconds based on our hyperparameter settings[2].

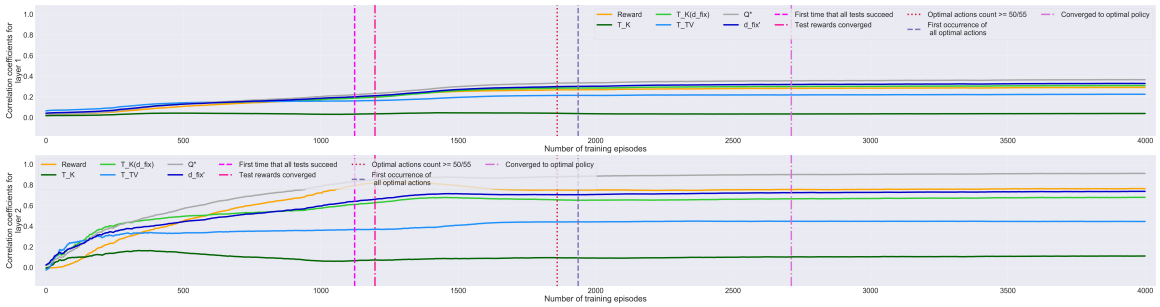


Figure C.6: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV}, c_{Q^*} and $c_{d'_{fix}}$ during training for both layers of a 2-layer DQN with a hidden layer size of 50 for FrozenLake 4x4 (OH).

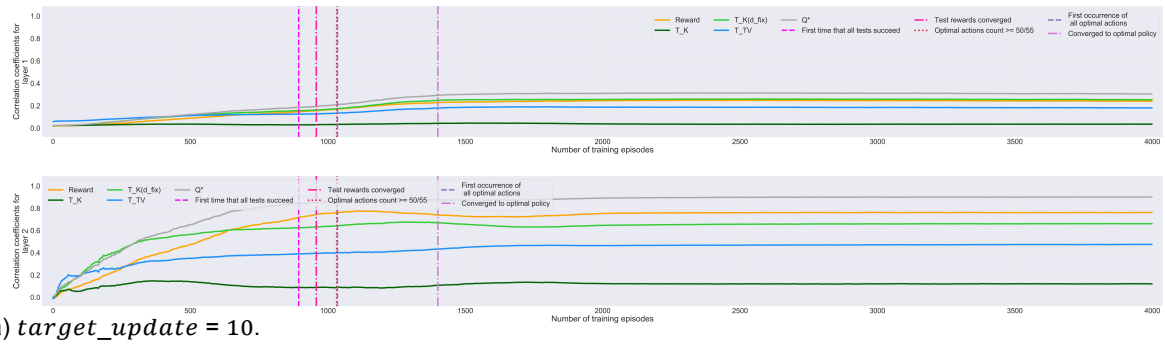
Figure C.6 shows that the value of $c_{d'_{fix}}$ is rather similar to the one of $c_{K(d_{fix})}$ in both network layers during training. Indeed, for the second layer of the 2-layer DQN, the average difference between the two measures is equal to 0.05 with a standard deviation of 0.02 and the largest difference occurs for training episode 394 with a value of 0.06. For the first layer, the mean difference is even lower with a value of 0.01 and a standard deviation of 0.01, whereby the largest arising difference is equal to 0.02. Therefore, computing $c_{d'_{fix}}$ is a viable alternative to calculating $c_{K(d_{fix})}$ for the FrozenLake 4x4 domain when it comes to evaluating how close to the coarsest Markov state representation an internal state representation is.

[2]Refer to Section D.1 for the hyperparameter values we use for the computation of d'_{fix} and to Section D.3 for information on how we run our computations.

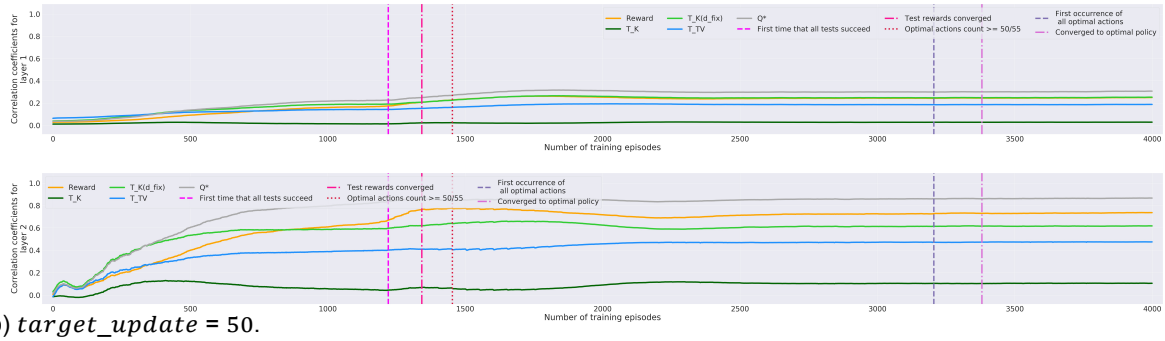
C.4. Sensitivity to Optimization Settings

In this section, we analyze the sensitivity of the results from Chapter 3 to the settings for the values of the hyperparameters for training the DQNs. The specific hyperparameters whose impacts we investigate are the frequency with which the target network is updated during training, the batch size, and the learning rate. We use 2-layer DQNs with a hidden layer size of 50 for FrozenLake 4x4 (OH) as example.

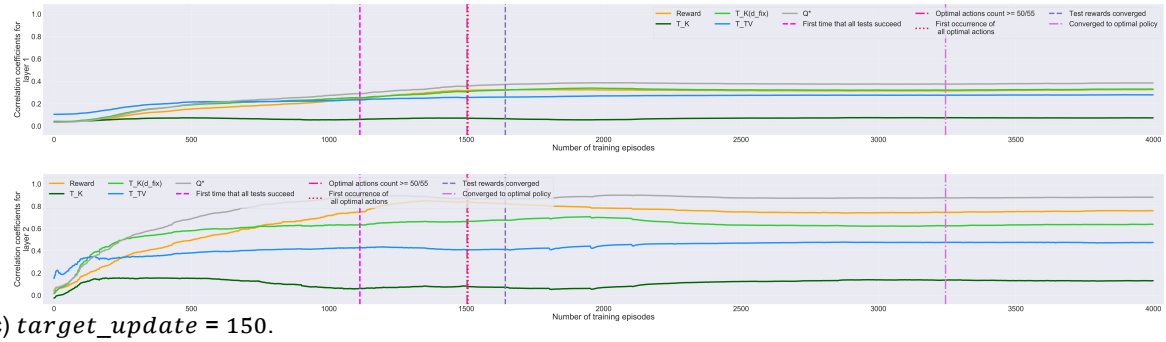
C.4.1. Target Network Update Frequency



(a) $target\_update = 10$.



(b) $target\_update = 50$.



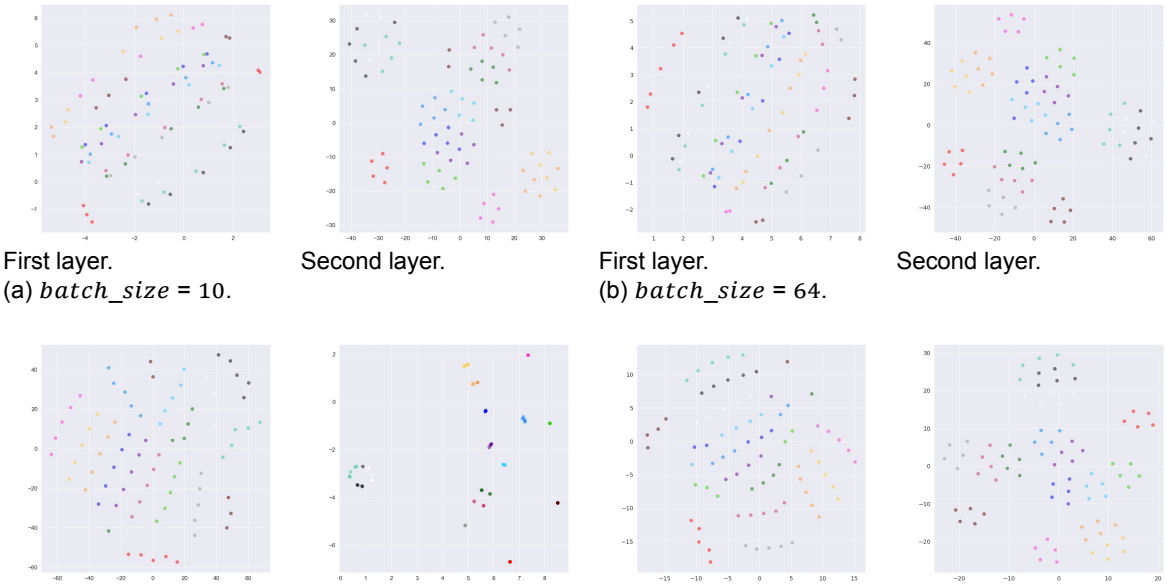
(c) $target\_update = 150$.

Figure C.7: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} during training for both layers of a 2-layer DQN for FrozenLake 4x4 (OH) when using different settings for the $target\_update$-parameter. The hidden layer size is equal to 50.

To rule out that the observed patterns for the correlation coefficients depend critically on the update frequency for the target network, we also trained 2-layer DQNs for FrozenLake 4x4 (OH) with values of 50 and 150 for the $target\_update$-parameter besides the original value of 10, while keeping all other hyperparameters fixed. As visualized in Figure C.7, changing the frequency with which the target network is updated during training significantly impacts neither the final values the correlation coefficients take on at the end of training nor the patterns of the correlation coefficients during training. The only difference is the speed with which the final first- and second-layer internal state representations are formed, with higher values for the $target\_update$-parameter leading to slightly later convergence to the final state representations. Since updating the target network every 10 episodes already ensures stable learning, it makes sense that updating the target network less often causes a later convergence to the true Q-values. This is the case, because more training episodes are required for the target network to output the Q-values based on a certain number of transitions if the target network is updated less frequently.

C.4.2. Batch Size

To explore the impact of using different batch sizes on the internal state representations that are formed and the way they are learned, we trained a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain with different batch sizes, while keeping all other hyperparameters fixed. Figures C.8 and C.9 depict that the batch size does not significantly impact the final output-layer state representation for a 2-layer DQN for FrozenLake 4x4 (OH). This means that all networks converge to the true Q-values equally well. Yet, the final first-layer representation has higher values for c_{Rew}, c_{TV}, c_{Q^*} and $c_{K(d_{fix})}$ for higher batch sizes. This can be explained by the fact that the first-layer representation groups states corresponding to the same ground state less closely together for smaller batch sizes (see Figure C.8). The reason likely is that using more transitions for updates results in the network being updated based on transitions with more different superfluous feature values for the ground states. Hence, states that differ solely in the superfluous feature value are grouped closer together at the beginning of training while the network has not yet converged to the Q-values. If transitions with fewer different superfluous feature values for states are employed for each update, states that correspond to the same ground state are moved less closely together at the beginning of training. Yet, due to the higher-than-necessary hidden layer size of 50, the DQN can still converge to the true Q-values.



First layer. Second layer. First layer. Second layer.
(a) $batch\_size = 10$. (b) $batch\_size = 64$.

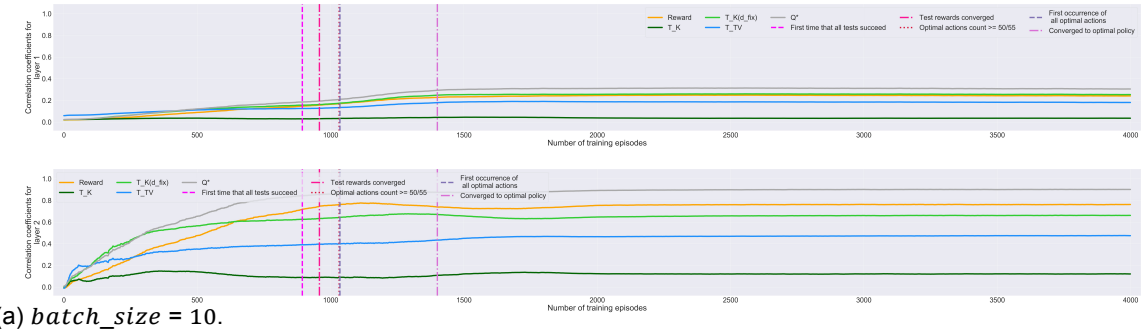
First layer. Second layer. First layer. Second layer.
(c) $batch\_size = 128$. (d) $batch\_size = 256$.

Figure C.8: t-SNE plots of the activations the states are mapped to at the end of training for both layers of a 2-layer DQN with a hidden layer size of 50 for FrozenLake 4x4 (OH) when using different batch sizes. Activations are computed for all states and every 5 states that differ solely in the superfluous feature and hence correspond to the same ground state are drawn in the same color.

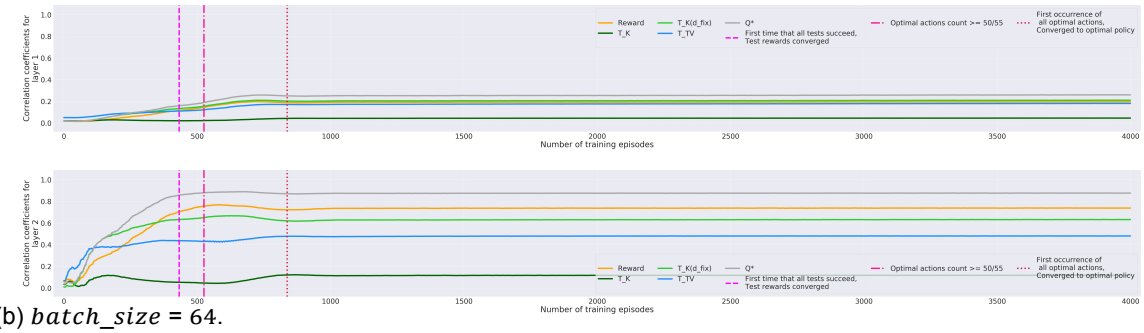
In addition, using a higher batch size causes some correlation coefficients in the second layer to first increase to a value higher than their final value and then to drop again. This, for example, occurs for c_{Rew} and $c_{K(d_{fix})}$ around training episode 500 (see Figure C.9). This can be explained by the fact that the DQN is trained to a higher extent and based on more accurate estimates of the n-step rewards by the target network after the target network has been updated $n - 1$ times for higher batch sizes, because more data is used during each update. For instance, recall that c_{Rew} is highest for a state representation that clusters states solely based on immediate rewards. Thus, if the DQN is trained to accurately predict Q-values based on n-step rewards for a low value for n, c_{Rew} for the second layer is higher than when the DQN accurately predicts the true Q-values.

C.4.3. Learning Rate

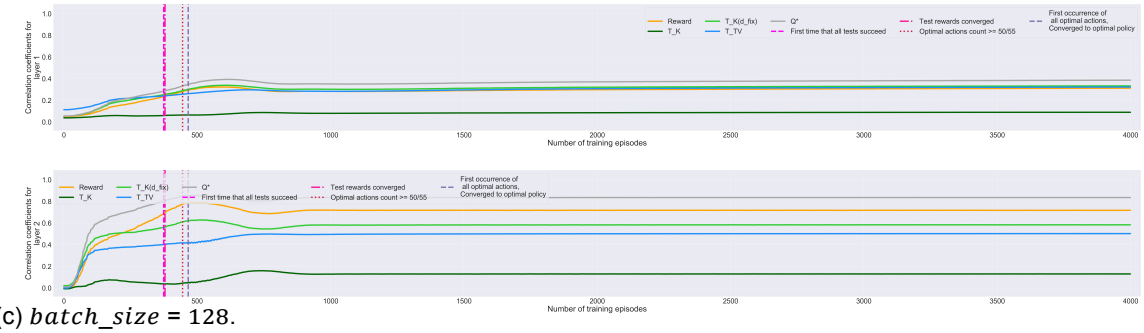
To investigate the impact of different learning rates on the internal state representations, we trained a 2-layer DQN with a hidden layer size of 50 for the FrozenLake 4x4 (OH) domain with learning rates of



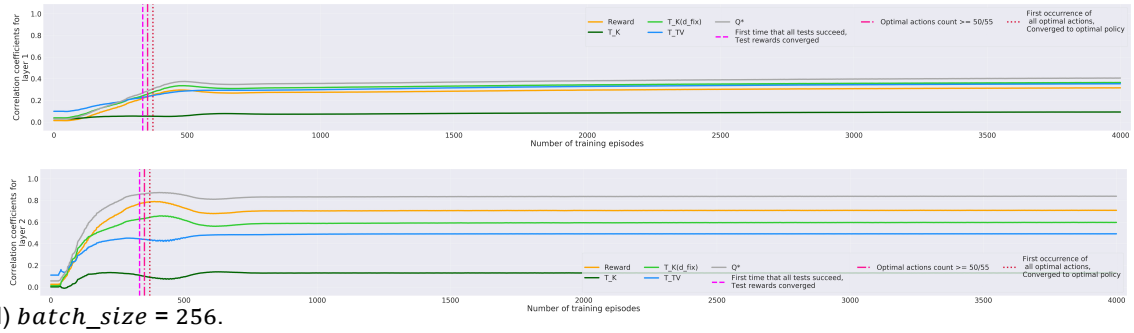
(a) $batch\_size$ = 10.



(b) $batch\_size$ = 64.



(c) $batch\_size$ = 128.

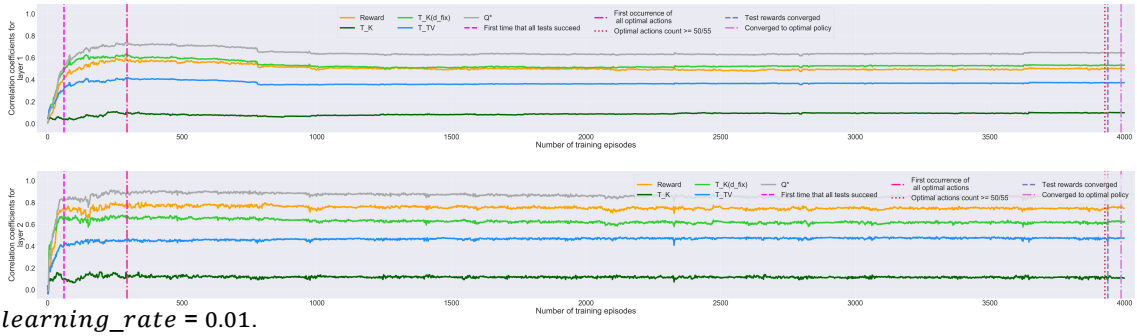


(d) $batch\_size$ = 256.

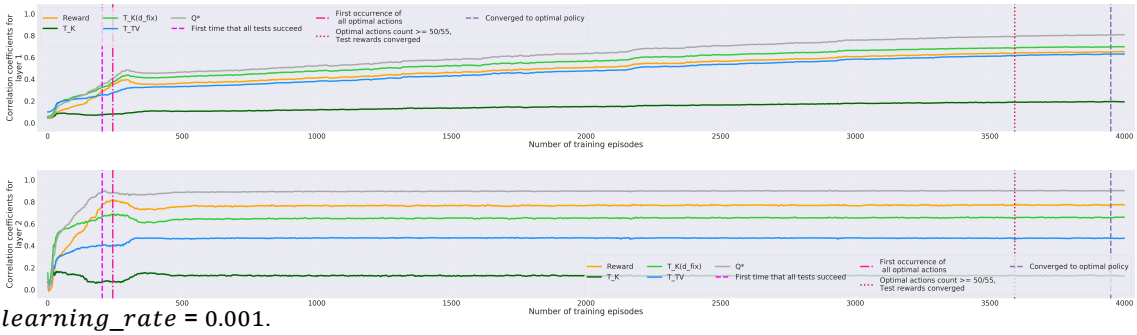
Figure C.9: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} during training for both layers of a 2-layer DQN for FrozenLake 4x4 (OH) when using different settings for the batch size. The hidden layer size is equal to 50.

0.01, 0.001 and 0.0001. Notice that the default learning rate for FrozenLake 4x4 (OH) is set to 0.0001 in our experiments (see Table D.1).

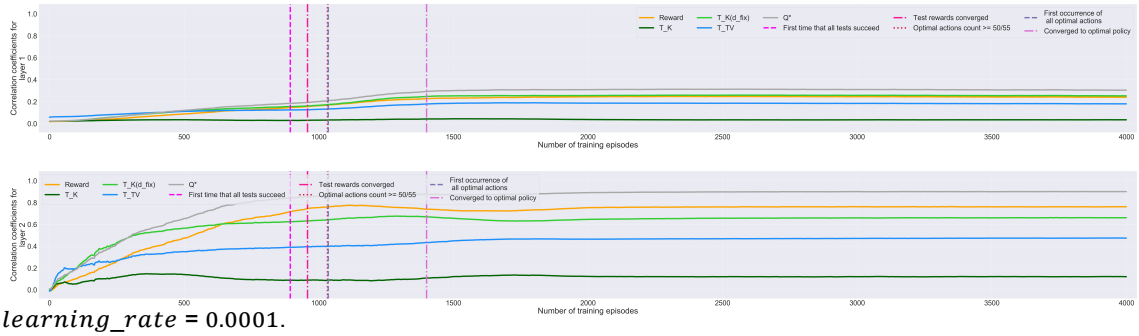
As visualized in Figure C.10, altering the learning rate for a 2-layer DQN for FrozenLake 4x4 (OH) does not significantly impact the learned second-layer representation. Thus, all tested learning rates allow the network to converge to the true Q-values equally well, albeit the correlation coefficients still fluctuate a lot for a learning rate of 0.01. The formed first-layer representation, however, strongly depends on the learning rate. More precisely, using a higher learning rate leads to higher values for $c_{K(d_{fix})}$, c_{Q^*}, c_{Rew}, and c_{TV} in the first network layer at the end of training for all sufficiently large hidden layer sizes (see Figure C.11). Figure C.12 shows that the reason for this observation is that higher learning rates enable the first-layer representation to learn to ignore the superfluous feature to a larger extent. This likely is the case, because higher learning rates cause even few occurrences of a certain



(a) $learning\_rate$ = 0.01.



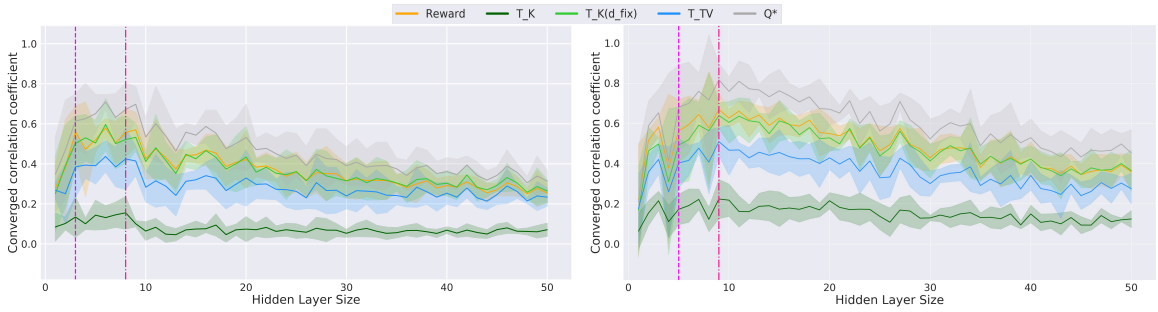
(b) $learning\_rate$ = 0.001.



(c) $learning\_rate$ = 0.0001.

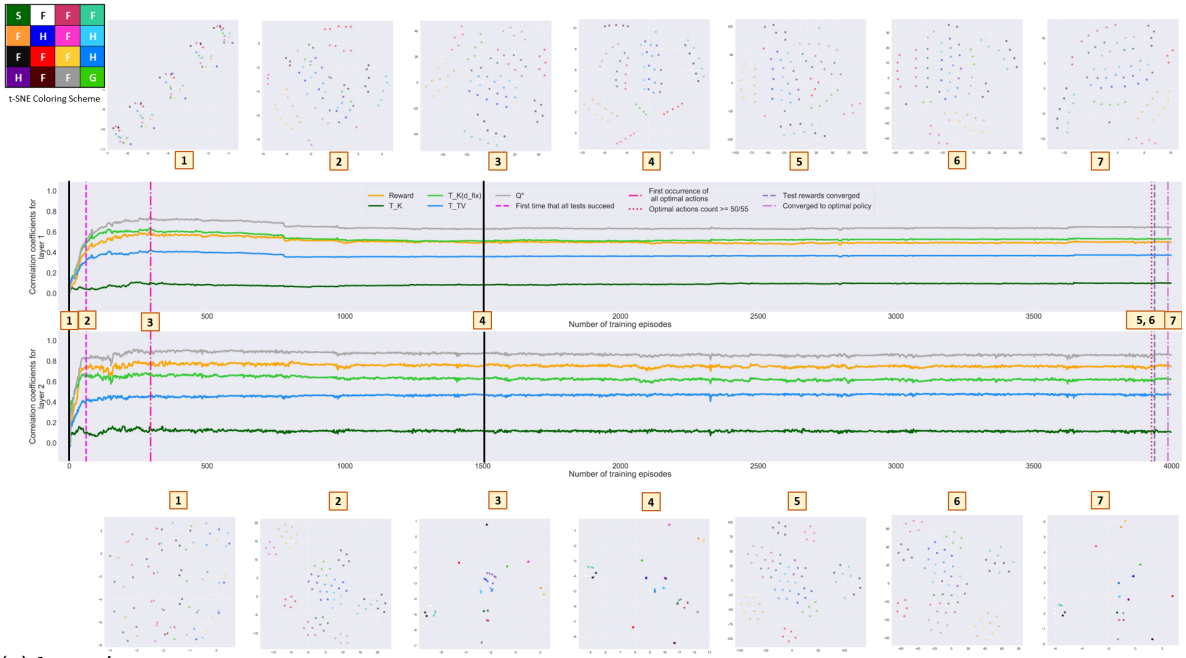
Figure C.10: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for different numbers of training episodes for both layers of a 2-layer DQN with a hidden layer size of 50 for FrozenLake 4x4 (OH) when using different settings for the learning rate.

superfluous feature value in the transitions used for updating the DQN to have a noticeable impact on the way the DQN is updated. This results in a stronger state space compression while the DQN has not yet converged than when a lower learning rate is used. The DQN can, however, also converge without this intensive state space compression occurring. Hence, for lower learning rates, the DQN converges before such a strong state space compression has been created in the first network layer. The impact of a higher learning rate thus is similar to the one of a higher batch size as discussed in Section C.4.2.

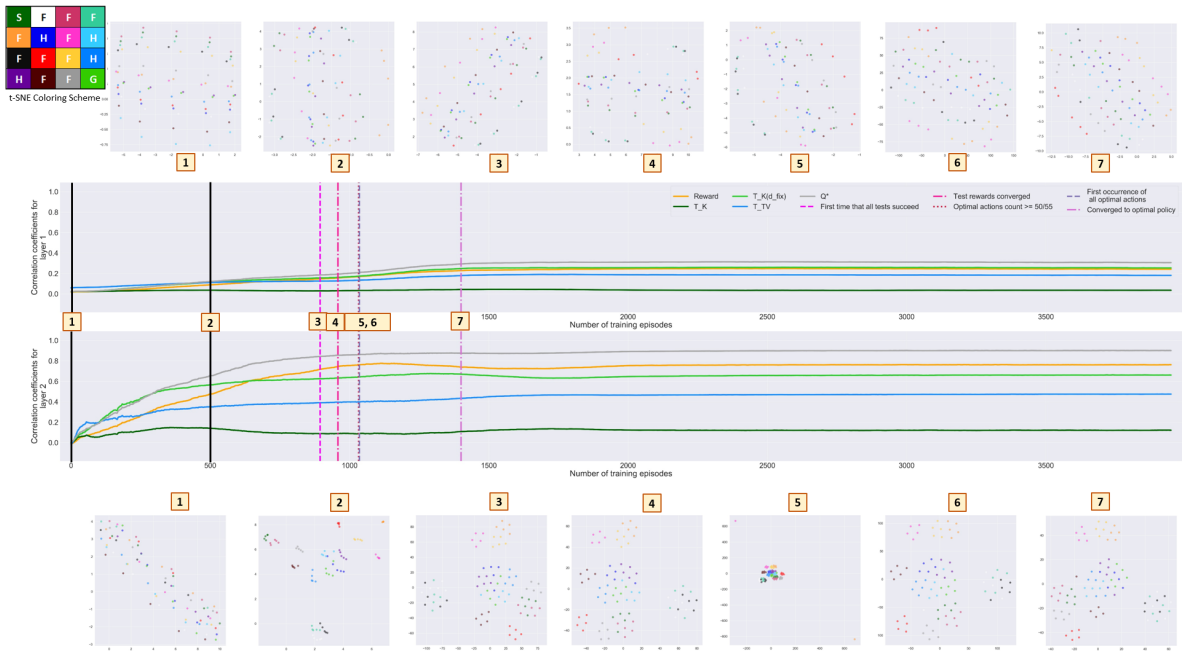


$learning\_rate$ = 0.0001. $learning\_rate$ = 0.01.

Figure C.11: Mean converged c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} in the first layer with 95%-confidence intervals for each hidden layer size for a 2-layer DQN for the FrozenLake 4x4 (OH) domain with different learning rates. Values are based on 5 repetitions. The two vertical lines indicate the lowest hidden layer sizes for which the test rewards converge and the network converges to the optimal policy, respectively, at least one out of 5 times.



(a) $learning\_rate$ = 0.01.



(b) $learning\_rate$ = 0.0001.

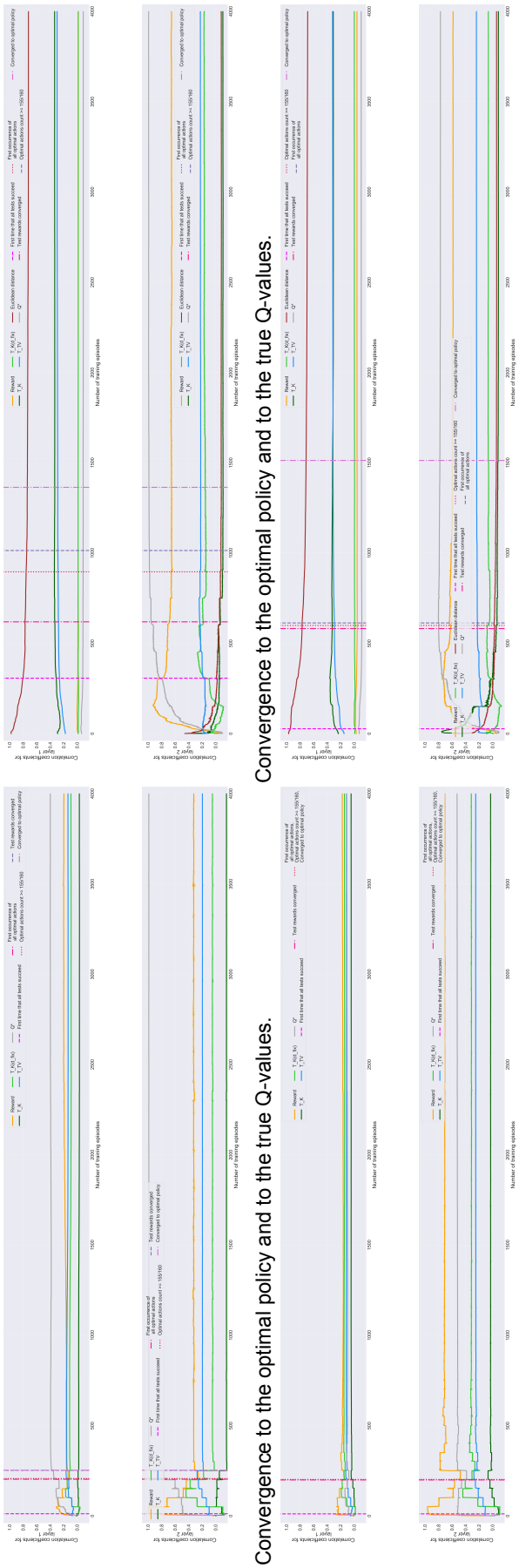
Figure C.12: t-SNE plots of the activations the states are mapped to and corresponding correlation coefficients in the layers of a 2-layer DQN with a hidden layer size of 50 for FrozenLake 4x4 (OH) throughout training for different learning rates. Activations are computed for all states and every 5 states that differ solely in the superfluous feature and hence correspond to the same ground state are drawn in the same color.

C.5. Convergence to the True Q-values vs. Solely Convergence to the Optimal Policy

A common problem when training DQNs is that the Q-values may diverge [1], whereby the network may still converge to the optimal policy. To explore whether convergence to the true Q-values is accompanied by different learned state representations in the network layers, we also trained 2-layer networks with a hidden layer size of 50 for the Gridworld 3x3 (OH) and Gridworld 3x3 (F) domains such that the Q-values diverge but the networks still converge to the optimal policy. Since both domains contain goal states with non-zero rewards, this can be achieved by not adding transitions that start at a goal state to the replay memory during training.

Figure C.13 visualizes that the learned first-layer representation for Gridworld 3x3 (OH) has a slightly higher value for $c_{K(d_{fix})}$ and a lower value for c_{Q^*} if the network solely converges to the optimal policy. Since the same trend can be observed in the second network layer, this reveals that the first network layer forms a state representation that is to a certain extent similar to the one in the last network layer if this is possible. Moreover, the learned first-layer representation for Gridworld 3x3 (F) does not change noticeably. Yet, this is due to the fact that for this form of state encoding, the first layer cannot group states together if and only if they have the same Q-values as discussed in Section 3.2.2.3. With respect to the last network layer for Gridworld 3x3 (F), Figure C.13 depicts a much higher value for c_{Q^*} and a lower value for c_{Rew} at the end of training when the network also converges to the true Q-values just as for Gridworld 3x3 (OH), as is to be expected for diverging Q-values.

Consequently, the representation created in the hidden layer is relatively insensitive to whether the network converges only to the optimal policy or also to the true Q-values. However, the learned first-layer representation is to a certain extent similar to the one formed in the last network layer if this is possible. This leads to a slightly different first-layer representation when the network only converges to the optimal policy when the ground state is one-hot encoded. How different the first-layer representation is thus largely depends on how different the formed second-layer representation is when convergence to the optimal policy but not to the true Q-values is achieved. Yet, this is only the case if the first layer can form a representation that is similar to the one in the second layer.



Convergence to the optimal policy and to the true Q-values.

Convergence solely to the optimal policy.

(a) Gridworld 3x3 (OH).

Convergence to the optimal policy and to the true Q-values.

Convergence solely to the optimal policy.

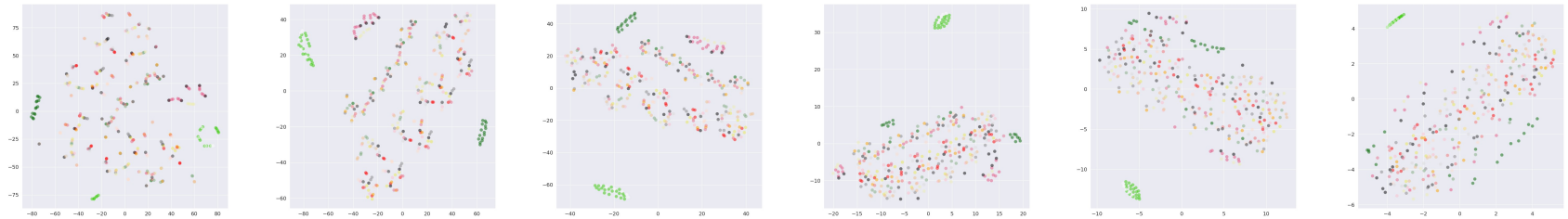
(b) Gridworld 3x3 (F).

Figure C.13: c_{Rew}, c_K, $c_{K(d_{fix})}$, c_{TV} and c_{Q^*} for different numbers of training episodes for both layers of 2-layer DQNs with a hidden layer size of 50 for Gridworld 3x3 (OH) and Gridworld 3x3 (F). The network converges either both to the optimal policy and the true Q-values or solely to the optimal policy.

C.6. Impact of the Perplexity on t-SNE Plots

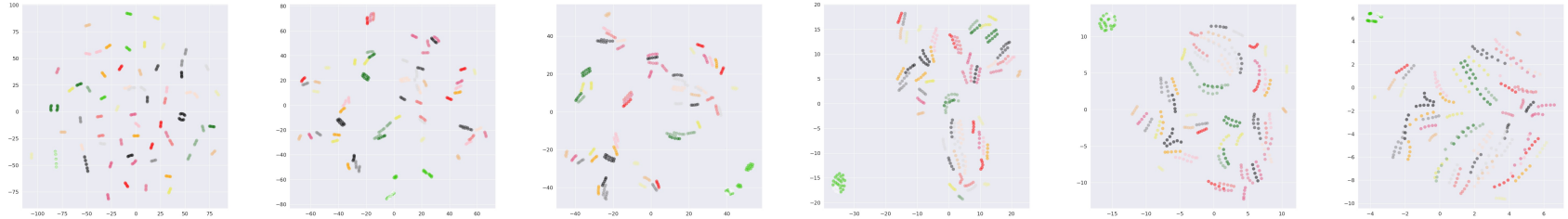
As described in Section 2.8, the output of the t-SNE algorithm depends on the perplexity parameter. Figure C.14 shows for the learned first-layer representation of 2-layer DQNs for Gridworld 3x3 (Aug) (OH) and Gridworld 3x3 (Aug) (OH)(N) which impact the perplexity has on the t-SNE plots. Recall that perplexity values between 5 and 50 are recommended by [40] and that the perplexity is set to 30 in our experiments. We make the following observations:

- A perplexity of 2 makes all clusters of activations look approximately equidistant and hence hides global structure in the data.
- When a hidden layer size of 6 is utilized, global structure appears in the t-SNE plots for perplexity values of 5, 10 and 30, and then becomes increasingly less visible for perplexity values of 50 and 100. Thus, a perplexity of 30 as is used in our experiments is appropriate for this state representation.
- For a hidden layer size of 50, all perplexity values greater than 2 lead to approximately similar results. Two main clusters are formed, whereby the larger one is composed of roughly equidistant clusters of activations belonging to non-terminal states and the smaller one of activations corresponding to terminal states. Since no perplexity value shows any global structure within the larger cluster, it is likely that such structure does not exist due to the larger hidden layer size and thus higher capacity of the network. However, it could also be that a different perplexity value is necessary or that simply no perplexity value exists for which this structure becomes visible [63]. Irrespective of whether such global structure cannot be shown or needs a very specific perplexity value, using a perplexity of 30 for this state representation is a reasonable choice given limited time to exhaustively search the space of perplexity values.



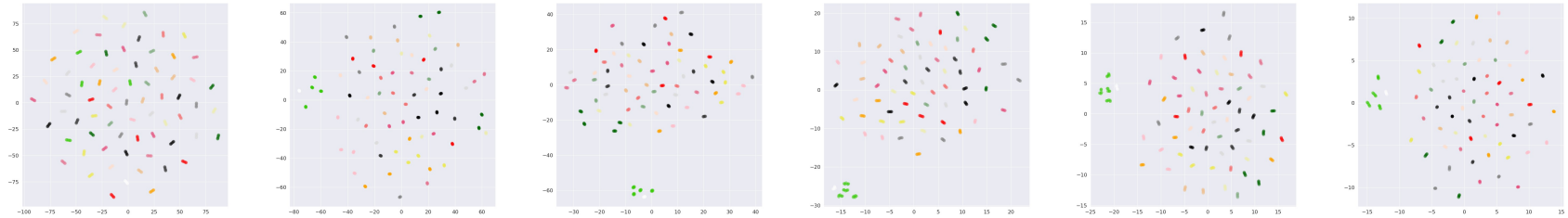
$Perplexity = 2.$ $Perplexity = 5.$ $Perplexity = 10.$ $Perplexity = 30.$ $Perplexity = 50.$ $Perplexity = 100.$

(a) Gridworld 3x3 (Aug) (OH), $hidden\_layer\_size = 6.$



$Perplexity = 2.$ $Perplexity = 5.$ $Perplexity = 10.$ $Perplexity = 30.$ $Perplexity = 50.$ $Perplexity = 100.$

(b) Gridworld 3x3 (Aug) (OH)(N), $hidden\_layer\_size = 6.$



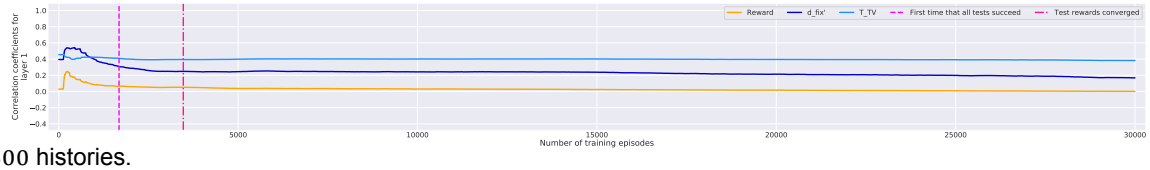
$Perplexity = 2.$ $Perplexity = 5.$ $Perplexity = 10.$ $Perplexity = 30.$ $Perplexity = 50.$ $Perplexity = 100.$

(c) Gridworld 3x3 (Aug) (OH)(N), $hidden\_layer\_size = 50.$

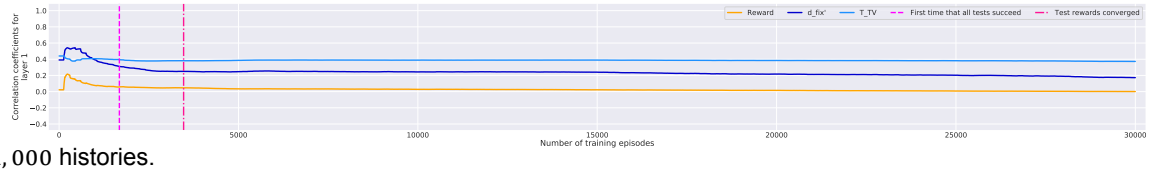
Figure C.14: t-SNE plots of the final first-layer state representations for Gridworld 3x3 (Aug) (OH) and Gridworld 3x3 (Aug) (OH)(N) for different values for the perplexity parameter of the t-SNE algorithm. Refer to Section 3.1.5 for the coloring scheme.

C.7. Correlation Coefficients based on Varying Numbers of Histories for POMDPs

Our experimental results for the internal state representations learned by DRQNs in Chapter B are based on computing the correlation coefficients based on a sample of 500 histories. As depicted in Figure C.15 for a DRQN with a hidden layer size of 32 for the Hallway domain, utilizing $1,000$ rather than 500 histories for the computation of the correlation coefficients leads to very similar results. Thus, the limited number of histories used to calculate the correlation coefficients does not pose a limitation to employing the correlation coefficients for an analysis of the internal state representations created by DRQNs for the Hallway domain.

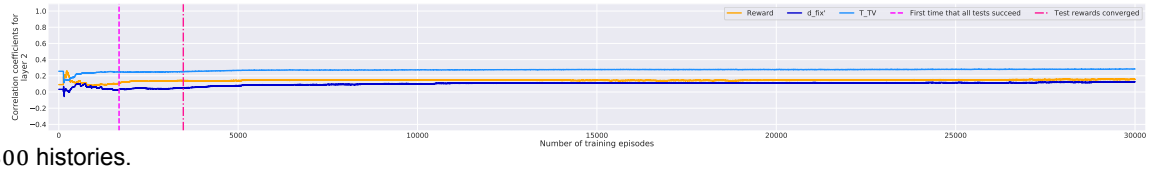


500 histories.

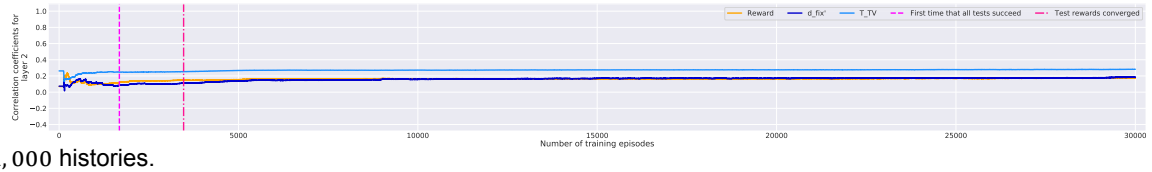


$1,000$ histories.

(a) First layer.

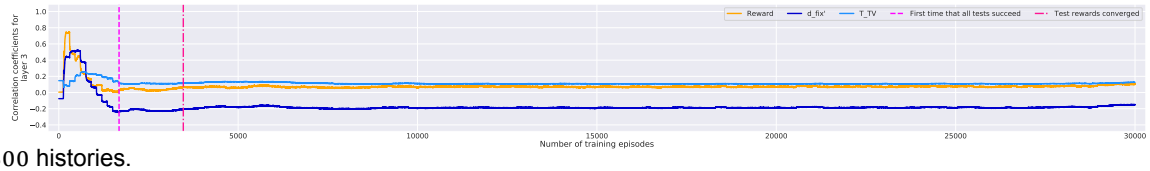


500 histories.

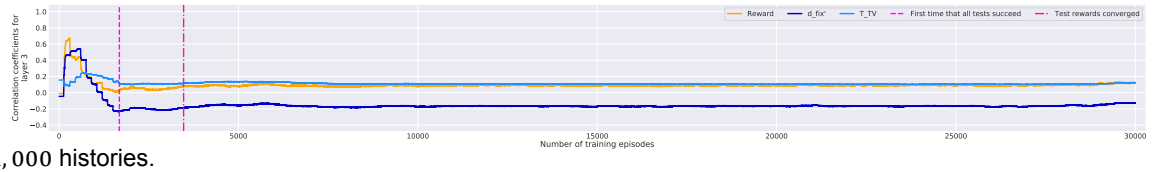


$1,000$ histories.

(b) LSTM layer.



500 histories.



$1,000$ histories.

(c) Output layer.

Figure C.15: c_{Rew}, $c_{d'_{fix}}$ and c_{TV} in each of the three layers of a DRQN for the Hallway domain during training. Both the first and the LSTM layer have 32 nodes. The correlation coefficients are computed based on the belief states resulting from sampling 500 or $1,000$ histories.

D

Implementation Details

Section D.1 contains details on the computation of correlation coefficients, Section D.2 describes the implementation and training of DQNs for our experiments, and Section D.3 states the technology used throughout our research.

D.1. Computation of Correlation Coefficients

This section provides information on the hyperparameters used for the approximation of d_{fix} via the algorithm by [8] in Section D.1.1 and the bisimulation-based correlation coefficient c_K in Section D.1.2.

D.1.1. Approximate d_{fix}

We approximate d_{fix} for belief states for POMDPs and for fully observable states for MDPs based on the approximation algorithm by [8] as described in Section 2.7.1.2[1]. We use a learning rate of 0.001, update the target network every 128 episodes, employ a batch size of 32, increment β by 0.00001 every 128 episodes up to a maximum value of 1, and train the network for $15,000$ episodes. Thereby, we utilize a neural network with one hidden layer and 32 nodes in this hidden layer.

D.1.2. Correlation Coefficient c_K

Definition. c_K is the Pearson correlation coefficient between the maximum Kantorovich distances T_K of the transition functions of the states on the one hand and the Euclidean distances between the activations the states are mapped to in a neural network layer on the other hand. We originally calculated c_K, because $c_{K(d_{fix})}$ is not independent of the reward function and of γ. However, computing the one-dimensional Kantorovich distance of two probability functions P and Q is equivalent to computing the cumulative distribution function of $|P - Q|$ [51]. Hence, calculating the one-dimensional Kantorovich distances of the transition functions has the caveat that the result depends on the number with which a state is encoded. For instance, if there is a domain with a single action a and three states s_1, s_2 and s_3, each encoded by their respective index, such that $P(s_1|s_1, a) = 1$, $P(s_3|s_2, a) = 1$ and $P(s_2|s_3, a) = 1$, then for a high value for c_K, the activation s_1 is mapped to should be closer to the activation s_3 is mapped to than to the one s_2 is mapped to. The reason is that the absolute distance between the encoded state numbers 1 and 3 is larger than the one between 1 and 2. The consequence is that the Kantorovich distance between the transition functions of s_1 and s_2 is 2, whereas the one between the transition functions of s_1 and s_3 is equal to 1. Therefore, while in general a high value for c_K indicates that there exist clusters in the space of activations for all states that have the same transition probabilities for all states, this is not necessarily the case for domains with few pairs of states with identical transition functions compared to the total number of state pairs. In such domains, c_K will be high only if there is a positive correlation between the Euclidean distance of the activations two states are mapped to and the absolute distances between the state numbers that they have non-zero transition probabilities to. Due to the dependence of c_K on the encoding of states and the fact that c_K no longer captures the notion of bisimilarity, we discontinued the computation of c_K after initial experiments.

[1] We use the PyTorch implementation by Miguel Suau de Castro of the algorithm by [8], which is in turn based on the implementation found here.

Computation. To calculate T_K for a pair of states s_i, $s_j \in S$ and an action $a \in A$, one has to compute the one-dimensional Kantorovich distance between the transition functions of s_i and s_j for a. When computing c_K, the maximum is taken over all actions. The pseudocode for this computation is given in Algorithm 4. To compute the one-dimensional Kantorovich distance, we utilize the wasserstein_distance[2] method from Python's Scipy package.

Algorithm 4 Computation of $max_{a \in A}(T_K(P(s_i, a), P(s_j, a)))$ for a pair of states $s_i, s_j \in S$

Require: S, A, $P(s_i)$ (transition function of s_i $\forall a \in A$), $P(s_j)$ (transition function of s_j $\forall a \in A$)

```
T_K_max ← 0
for a ∈ A do
    T_K_max ← max(T_K_max, Wasserstein_distance(P(s_i, a), P(s_j, a)))
end for
```

D.2. DQN Implementation and Training

For each fully observable domain, a DQN is implemented in PyTorch 1.3.1 [47] and trained by minimizing the loss given in Equation 2.5 via the Adam algorithm [34][3]. Thereby, a target network and a replay memory are used to improve learning stability as described in Section 2.4.1. For each domain, two different network architectures are tested, one with two layers and one with four layers. The ReLU activation function is used for all non-terminal layers, which is defined as follows:

$$ReLU(x) = max(0, x).$$

Each hidden layer has the same size, which is given by the $layer\_size$ parameter in Tables D.1 and D.2, unless an experiment involves changing the hidden layer size. The hyperparameter values are determined via grid search for each domain and the final values are listed in Table D.1 for the FrozenLake domains and Table D.2 for the Gridworld domains.

The policy of the agent during training is ϵ-greedy, which means that the agent takes the greedy action with probability $1 - \epsilon$ and a random action with probability ϵ. In case ϵ is decayed over time, its value is determined by the following equation:

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \times exp(-decay\_rate \times num\_steps), \tag{D.1}$$

where ϵ_{min} is the minimum value for ϵ, ϵ_{max} is the maximum value for ϵ, $decay\_rate$ is the decay rate for ϵ and num_steps is the number of steps taken so far during training.

D.3. Technology

In the following, we describe the technology utilized throughout our experiments.

D.3.1. HPC Cluster

Computations are run on CPUs of the HPC cluster of the Intelligent Systems Department at the Delft University of Technology.

D.3.2. Min-Cost-Flow-Class

The Min-Cost-Flow-Class is a C++ library that encompasses several solvers for minimum cost flow problems. To compute the bisimulation metric component $T_k(d)$, we utilize the MCFZIB solver [2] from the Min-Cost-Flow-Class just like the original work on bisimulation metrics does [14].

D.3.3. Python

Python is an interpreted high-level programming language that follows standards from procedural, imperative to object-oriented, and functional programming. In addition, it contains multiple useful packages, including Matplotlib [28] for creating effective data visualizations, NumPy [60] for efficient vectorized computations, OpenAI Gym [7] for RL research, PyTorch [47] for machine learning and especially

[2] The Kantorovich distance is also called Wasserstein distance.

[3] The implementation of the DQNs is based on a DQN implementation for the CartPole task by Adam Paszke [46] and a Q-learning tutorial by Tiew Kee Hui [27].

Table D.1: Hyperparameter values for DQNs trained for the FrozenLake domains.

Parameter	Description	FrozenLake 4x4 (OH)	FrozenLake 4x4 (F)	FrozenLake 8x8
batch_size	Batch size for training	10	16	128
decay	Whether to decay ϵ	True	True	True
decay_rate	Decay rate for ϵ	0.0001	0.0001	0.00001
discount_factor	Discount factor of MDP	0.9	0.8	0.99
layer_size	Size of hidden layers	50	50	50
learning_rate	Learning rate α	0.0001	0.005	0.0005
lr_decay	Whether to decay α	False	False	False
lr_decay_rate	Decay rate for α	/	/	/
max_epsilon	Starting value for ϵ	1	1	1
max_steps	Max. # of steps per episode	200	200	200
min_epsilon	Min. value for ϵ	0.01	0.2	0.1
num_episodes	Number of training episodes	4,000	4,000	10,000
replay_memory_size	Capacity of replay memory	50,000	50,000	50,000
target_update	Update frequency for target network	10	10	150

Table D.2: Hyperparameter values for DQNs trained for the Gridworld domains.

Parameter	Description	Gridworld 3x3 (OH)	Gridworld 3x3 (F)	Gridworld 3x3 (Aug)	Gridworld 5x5
batch_size	Batch size for training	128	128	128	128
decay	Whether to decay ϵ	True	True	True	True
decay_rate	Decay rate for ϵ	0.0001	0.0001	0.00001	0.0001
discount_factor	Discount factor of MDP	0.85	0.75	0.99	0.85
layer_size	Size of hidden layers	50	50	50	50
learning_rate	Learning rate α	0.005	0.001	0.00001	0.005
lr_decay	Whether to decay α	False	False	False	False
lr_decay_rate	Decay rate for α	/	/	/	/
max_epsilon	Starting value for ϵ	1	1	1	1
max_steps	Max. # of steps per episode	100	100	100	100
min_epsilon	Min. value for ϵ	0.1	0.2	0.3	0.01
num_episodes	Number of training episodes	4,000	4,000	30,000	4,000
replay_memory_size	Capacity of replay memory	50,000	50,000	50,000	50,000
target_update	Update frequency for target network	50	150	150	100

deep learning, and Scikit-learn for machine learning tasks in general [48]. Version 3.7 of Python is used in this work.

D.3.3.1 NumPy Library

NumPy is a Python library that enables high-performance numerical calculations by reducing the number of operations, vectorizing computations, and avoiding the copying of data into memory. Version 1.16.5 of this library is utilized in this work.

D.3.3.2 OpenAI Gym

Version 0.15.4 of the OpenAI Gym library is utilized in the experiments in this work. For the experiments for MDPs, we make use of the implementation of the 4x4 and 8x8 FrozenLake domains. Thereby, we alter the original implementation to allow for modified reward and transition functions for our experiments in Section 5.1, which involve the transfer of trained DQNs to related domains. Moreover, we implemented the Gridworld 3x3 (Aug) domain, the variably sized Gridworld domain used in the work of [14], and the Hallway domain [38] based on OpenAI Gym.

D.3.3.3 PyTorch Library

PyTorch is an open-source Python library that performs dynamic tensor computations combined with automatic differentiation and GPU acceleration. PyTorch achieves high levels of flexibility and performance while maintaining ease of use for the user. Version 1.3.1 is employed in this work to implement the neural networks.

D.3.3.4 Scikit-learn

We employ version 0.21.3 of the Scikit-learn library to visualize the activations in neural network layers via the t-SNE algorithm described in Section 2.8. Moreover, we utilize the K-Means algorithm to cluster histories based on bisimulation-based pairwise distances for t-SNE plots for partially observable domains in Chapter B.