DOI: 10.1111/cgf.14839
Eurographics Conference on Visualization (EuroVis) 2023
D. Archambault, R. Bujack, and T. Schreck
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 3

VISITOR: Visual Interactive State Sequence Exploration for Reinforcement Learning

Yannick Metz[1], Eugene Bykovets[2], Lucas Joos[1], Daniel Keim[1], Mennatallah El-Assady[2]

[1]University of Konstanz [2]ETH Zürich

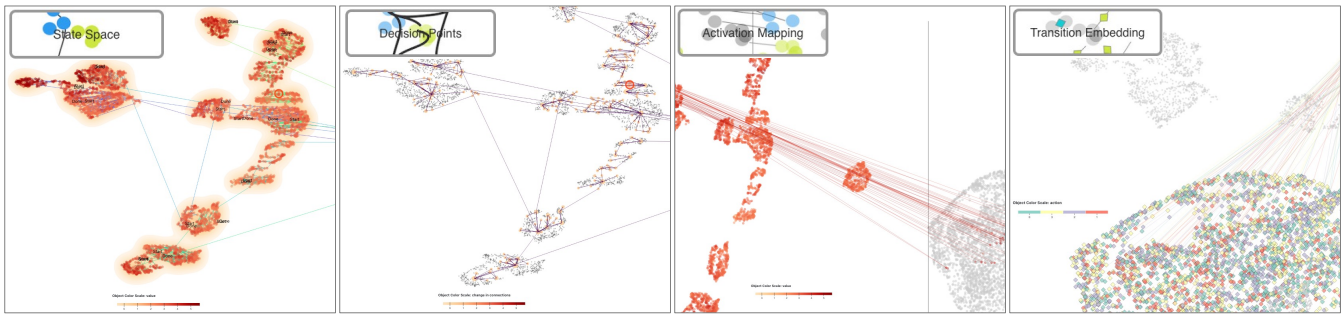

Figure 1: *Visualizing state-action sequences in VISITOR: Overlaying a projection with custom visual encoding to explore different aspects: (a) State space embeddings with semantic zoom and state aggregation, (b) bundling of state-action sequences with highlighted decision points that emphasize the divergence of episodes, (c) visual mapping of states and network activations, (d) embedding of state-to-state transitions.*

Abstract

Understanding the behavior of deep reinforcement learning agents is a crucial requirement throughout their development. Existing work has addressed the identification of observable behavioral patterns in state sequences or analysis of isolated internal representations; however, the overall decision-making of deep-learning RL agents remains opaque. To tackle this, we present VISITOR, a visual analytics system enabling the analysis of entire state sequences, the diagnosis of singular predictions, and the comparison between agents. A sequence embedding view enables the multiscale analysis of state sequences, utilizing custom embedding techniques for a stable spatialization of the observations and internal states. We provide multiple layers: (1) a state space embedding, highlighting different groups of states inside the state-action sequences, (2) a trajectory view, emphasizing decision points, (3) a network activation mapping, visualizing the relationship between observations and network activations, (4) a transition embedding, enabling the analysis of state-to-state transitions. The embedding view is accompanied by an interactive reward view that captures the temporal development of metrics, which can be linked directly to states in the embedding. Lastly, a model list allows for the quick comparison of models across multiple metrics. Annotations can be exported to communicate results to different audiences. Our two-stage evaluation with eight experts confirms the effectiveness in identifying states of interest, comparing the quality of policies, and reasoning about the internal decision-making processes.

CCS Concepts

• ***Human-centered computing*** → *Visual analytics;* • ***Computing methodologies*** → *Reinforcement learning;*

1. Introduction

Deep reinforcement learning (RL) is widely recognized as an instrumental and promising approach to enable sequential decision-making applications [ADBB17, MBP*23], e.g., in gameplay [MKS*13, SHS*17] or robotics [SMG22]. However, the adoption to real-world applications is challenging due to the difficulty of training [Irp18, DLM*20, HIB*], as well as the inherent opaqueness and unpredictability of agents.

The complex training and evaluation process of Deep RL agents can benefit from interactive visual workflows [MSS*], e.g., by supporting the analysis of agent behavior and increasing interpretability [WGSY19, HLB*20, MSHB22, EHA*22]. Many problems, in particular when considering the deployment of trained agents, are connected to the black-box nature of deep learning models; not only is the decision-making opaque but also the behavior of agents trained via RL is oftentimes unpredictable. To address this lack of transparency in understanding the behavior and decision-making of AI models, the area of explainable AI has recently experienced a surge in research [AB18]. However, compared to tasks like image classification, sequential decision-making places an additional burden on XAI techniques. Due to the correlation between subsequent steps, it is not sufficient to explain single decisions in isolation, for example, with single input attributions [SVZ13]. Instead, explana-

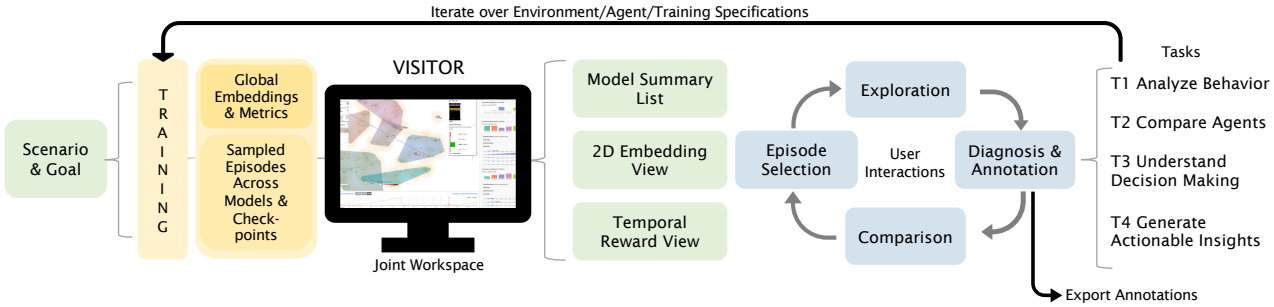

Figure 2: *The workflow of VISITOR: The scenario sets the general framing, shaping design decisions and expectations about training outcomes. To allow for analysis at scale, we compute global metrics and embeddings based on all available data and then visualize a sampled subset of episodes for the analyst. These episodes are visualized in a joint workspace, consisting of a 2D state sequence embedding view, a reward, and an environment action probability view. These views facilitate a process of iterative exploration, annotation, comparison, and decision-making.*

tions must always be contextualized in the sequential process. A second factor complicating the generation of meaningful explanations is the uncertainty about the distribution of states that are available for analysis and generation of explanations. In non-sequential tasks, a fixed test set is generally available. During the evaluation of RL algorithms, the user has no control over the states for which explanations can be generated.

In related work, interpretable and explainable deep reinforcement learning is mainly covered by visual analytics systems, visualizing state-action sequence and behavior [WGSY19, HLB*20, MSHB22], as well as input attribution methods [GKDF18, NIAN]. Sequence-focused approaches help to better understand the progression of states and, therefore, the distribution of samples available for explanations. However, they often cannot provide contextualized explanations for decisions. Conversely, input attributions, particularly when only applied to single states, cannot communicate actual behavior and the resulting sequence of observations to the user.

In this paper, we present VISITOR, a visual analytics approach for interactively exploring the state sequence space of RL agents. In Figure 2, we summarize the entire workflow, from the design phase and data collection to analysis. A potentially large set of episodes across checkpoints and models is collected during training or evaluation. A human expert can select and explore relevant episodes. In Section 3, we define four user tasks that VISITOR is designed for. The system, including the workflow and elements, are presented in Section 4. Our approach builds on existing work in interactive sequence data visualization [WGSY19, WZY*21, HSH*21] and state embedding visualizations (e.g., utilizing t-SNE) [ZZM16, MKS*13]. However, existing work is not targeted at the analysis and comparison of multiple policies. By enriching these embeddings with automated annotations, filtering, and highlighting, as well as dynamic multi-scale interactions, we achieve a compact and highly flexible exploration tool for sequential decision-making with deep reinforcement learning.

Beyond increasing the users' trust in the system, analyzing the behavior of trained agents is crucial during the validation and optimization of the RL environment and training settings. Understanding how agents either fail or succeed can inform design choices and even drive future algorithm developments. Therefore, in this paper, we present an approach that enables the comparison of different agents and allows analyzing an agent's behavior during training.

In addition, since the reward function is the core optimization objective in RL, it should play a central role in the analysis of an agent's behavior. Hence, this paper also describes the coupling of state sequence embeddings with an interactive reward view.

Overall, this paper contributes: (1) a general approach to jointly analyze behavior and internal representations of Deep RL agents, with a particular focus on the comparison of agents at different levels of detail, (2) a implementation of multiple linked views, including a novel RL state sequence embedding with multiple layers, (3) an expert study evaluation with eight RL experts confirms our approach's utility for analyzing the behavior of several RL agents in three different environments. Our technique and visual elements showcase possible solutions to deal with large, unlabeled state and state-action spaces beyond the domain of reinforcement learning.

2. Background and Related Work

In the following, we give an overview of the key concepts of (deep) reinforcement learning before reviewing related literature work. As our framework is generalizable to any RL algorithm, we keep the discussion of prerequisites on a general level.

2.1. Prerequisites for Deep Reinforcement Learning

In reinforcement learning, an **agent** perceives and acts autonomously in an **environment** while pursuing a specified goal. The agent acts in an environment that has a state $s_t \in \mathcal{S}$ at time step t, where $\mathcal{S}$ is the set of all possible states. The framing is formalized in the Markov Decision Process (MDP) framework [SB18]. Note that in most scenarios, especially in the real world, the state of the environment is not fully observable. Reasons for this could be occlusions, missing sensor capabilities, and others. We refer to the part of the state that is accessible to an agent at time step t as an **observation** $O_t \in \mathcal{O}$. The agent can perform an **action** $a_t \in \mathcal{A}$ in the environment and observe the effect of the action in the next step. The agent has access to a scalar **reward** signal r_t at each step, which encodes the agents' performance in achieving the desired goal. The reward signal can be delayed, which means it does not have to be informative at each step [SB18]. Instead, a high reward value may only be a sparse signal, e.g., when reaching a goal. An RL agent learns to maximize the accumulated reward in a trial-and-error process, i.e., trying out actions and observing the resulting reward. The goal of RL is to learn a **policy** $\pi : \mathcal{O} \rightarrow \mathcal{A}$ that controls the behavior of the agent. The future expected reward for a given policy, called the **value** $v(s)$ of a state, plays an essential role in many RL algorithms, as it denotes

the predicted *worth* of a particular state for a trained agent.

Deep RL combines traditional RL algorithms with deep neural networks, facilitating their application to unstructured, potentially high-dimensional observations, e.g., image sequences. Because of this, Deep RL is affected by many of the issues of deep learning as a whole, particularly the black-box nature of models.

2.2. Visual Analytics for Explainable Deep RL

Architectures for Interpretable RL [HCDR21] often rely on human-readable rules or are based on deep learning architectures that incorporate information structures to learn human-interpretable representations [MMSV20, LSSP21, AS19, HAB*20, PCC*, NIAN, SHSW21]. In this work, we provide tools to explain the decisions and investigate internal representations of arbitrary policies in RL.

Explainable AI for RL – Apart from visual analytics-based solutions, there have been additional efforts to increase the explainability of reinforcement learning agents. This included fixed projections of the state space [ZZM16], utilization of input attributions [GKDF18, SHS*22], and sample-based explanations [SG20]. We take inspiration from this work but combine it with methods from visual analytics to enable the analysis of highly complex state sequences and a larger set of user tasks. We provide a more detailed discussion of related work in the supplementary material.

Visualizing State Space With Projections – Using 2D projections to visualize sequences of arbitrary states is an established approach [BSH*15]. Recently, both *projection path explorer* [HSH*21] and *projection space explorer* [EHA*22] have presented visualizations of decision-making processes via two-dimensional projections, using tools like clustering of states, or glyph markers to highlight structures in the data. While we share the basic tool of projection-based state sequence visualizations, we embed this visualization in a system targeted at RL debugging, addressing specific challenges via different layers, annotation tools, and components facilitating a workflow targeted at RL.

Agent Movement Patterns – *DQNViz* [WGSY19] is a visual analytics application to analyze the behavior of agents in Atari environments (see Arcade Learning Environment [BNVB13]). Its trajectory view shows movement patterns of the pad in breakout controlled by an agent during training. However, the presented approach has a challenge in generalizing to environments beyond the targeted ones. It only supports one-dimensional movement patterns and discrete action spaces. In contrast, our approach is suitable for arbitrary types of behavioral patterns and enables comparison between models.

Analysis for Recurrent Architectures – *DRLViz* [JVW20] is an approach to interpret the internal (latent) memory of an agent as found in LSTMs [HS15, HS97] or other recurrent neural networks. The main view is a memory timeline, which visualizes how the values of the hidden state vector change. The tool shows how certain values in the hidden state can be linked to objectives like collecting an item at a game level. *DRLIVE* [WZY*21] analyzes RNN-based deep RL agents. The work shares our ambition to enable the analysis of longer episodes. The work also utilizes reward line charts and embeddings of different features, such as the pixel values of a game screen or the hidden state of a cell in an RNN. Both approaches rely on a memory-based architecture (RNNs) with a relatively low-dimensional hidden state vector and make multiple assumptions to enable their visual design. In contrast, our method is agnostic to the type of environment and architecture and does not rely on a sequential structure of recurrent activations to represent state sequences.

Domain-Specific Analysis – *DynamicsExplorer* [HLB*20] diagnoses a trained policy for a robot control task. A robot arm has to move a small ball in a maze by tilting a platform. The tool incorporates some views to track the trajectories of the ball in the maze during episodes. Similar to DLRViz, there is a condensed temporal visualization of the hidden state of a recurrent neural network. The tool enables the inspection of real-world conditions over test runs and tries to explain the relation between simulation parameters and internal model parameters. It clusters similar trajectories to identify commonalities between runs. Again, the authors note the difficult generalization of the approach to other use cases.

High-Level Aggregation of Agents – Saldanha et al. [SPBA19] showcase a tool that helps data scientists during experimentation. The tool provides insight into hyperparameter settings, supports identifying which behaviors lead to higher or lower rewards, and how the behavior of agents evolves during training. It includes a view aggregated the trajectories of an agent over an episode. Furthermore, a thumbnail is computed to represent an agent's trajectory performance in 2D space.

In summary, most of the existing work either targets or requires a custom architecture [JVW20, AS19, VMS*18, WZY*21] or is currently only applicable to a very specific use case or environment [WGSY19, JVW20, HLB*20]. In contrast, we present a general approach specifically tailored to reinforcement learning and supporting arbitrary RL environments and algorithms.

3. System Requirements and Tasks

Based on interviews with RL experts, we present two particular RL explainability challenges, followed by the derived requirements and user tasks to address these challenges.

Exploring the emerging state space – In RL, we face a special situation compared to supervised/unsupervised learning. When analyzing a standard ML model, e.g., for image classification, we have full control over the samples included in an evaluation set $\mathcal{S}_{eval}$. For sequential decision-making tasks, we only have full control over the start states, while the subsequent states depend on the learned policy and are, therefore, not controllable. This means that the state space we can analyze an agent over is emergent and depends on the policy π, start state s_0, and environment's transition probabilities $\mathcal{P}$. In supervised learning, we can approximate global explanations by aggregating local ones, e.g., for whole dataset classes. In RL, there is no obvious aggregation function for states that could anchor explanations. Therefore, one of our core objectives is to provide a structured and globally-consistent visualization of the state sequence space.

Duality of Observable Behavior & Internal Representations – The state sequence space represents the *observable/behavioral view* of an agent. Explanations of agent behavior have to be contextualized in this trajectory space. The hidden *internal state representations* determine how the agent acts in the environment. Therefore, understanding the internal representations is a priority to achieve

explainability. In a deep RL agent, the state is represented in the activations of one or multiple of its neural networks. Instead of the observable state-action sequences, these internal representations create activation-action sequences. To tackle this challenge, we enable the joint visualization of observable state sequences $s_0, s_1, ... \in \mathcal{S}$ together with latent states $z_0, z_1, .. \in \mathcal{Z}$. Based on RL-expert feedback, we chose a high abstraction level to analyze internal representations; instead of, e.g., displaying absolute activation values, we focus on the similarity of internal representations.

3.1. Deriving User Tasks from Expert Interviews

Our approach is primarily targets experts in deep– and reinforcement learning. We derived a set of common tasks from an initial internal draft of user tasks in agreement with surveyed RL experts. These external experts have a multi-year experience in the design and training of RL agents and were part of the expert study we present in a later chapter. In a semi-structured interview, we asked the experts about their existing workflow for the training and evaluation of RL agents. Specifically, we asked about perceived gaps in the existing workflows and requested a high-level description of potentially interesting aspects that experts would consider during analysis. The interviews revealed that evaluation and analysis of trained agents mainly focus on two aspects: (1) metrics-based evaluation, e.g., based on achieved accumulative reward, training loss, or task-specific metrics, and (2) analysis of the agent's behavior, mostly via visual inspection. Metrics-based evaluation is covered by existing visualization tools like *TensorBoard* [Ten], but due to the demand of experts, we include summarized metrics to enable a quick overview and comparison between different models.

In contrast, behavioral analysis is underdeveloped and often restricted to mere visual inspection of videos. Furthermore, none of the experts regularly uses XAI tools like attributions maps. Based on their experiences, the experts generally did not expect vast benefits from analyzing the individual values of hidden representations, such as network activations. While input attributions were generally rated valuable, doubts were raised about their validity. Also, attribution maps were only used to analyze a few special cases. Lastly, a multitude of different RL environments and algorithms were used among our participants. Hence, tools for explainable RL need to cover a broad range of possible scenarios, relying on the basic conceptual framing of reinforcement learning (as presented in Section 2).

These interviews aimed to validate a set of user tasks and functional requirements. Therefore, these interviews were conducted before introducing the proposed visual analytics workflow to the expert. Based on these observations, we derive a set of requirements: The tool should integrate well with existing metric-based evaluation tools and serve as an extension of established workflows to the domain experts. It should enable behavioral analysis on different levels of detail, allowing for detailed diagnosis of individual decisions on demand. Internal representations should be visualized on a high level of abstraction, i.e., the inspection of individual network activations was of low importance to the experts. Lastly, the tool must enable simple switching between different models and checkpoints. Based on analyzing these requirements, we identify the following four tasks:

(T1) Analyzing high-level behavior of trained agents, with the option of detailed diagnosis. An expert must be able to capture the behavior across whole episodes. In particular, they are interested in spotting behavioral patterns that persist between multiple episodes or special cases and outliers. The application must, therefore, also provide multiscale interactions to investigate behavior in smaller sub-sequences. Lastly, the expert might be interested in additional metrics and statistics to evaluate the quality of agents.

(T2) Compare policies between runs & different training stages. Experts can compare agents to previous versions and other configurations and designs. Analyzing different checkpoints enables model developers to understand the training process and find reasons for stagnation or failure. Comparison can guide developers and put an agent's performance into context.

(T3) Understanding the reasons behind an agent's decision-making. This includes an interpretable view of a policy's internal representations at different scales. The expert must be able to spot overlapping and diverging internal representations of states. They should be able to investigate "junction points" in detail, e.g., by closely analyzing input attributions affecting single decisions and understanding which factors might lead to different decisions.

(T4) Get actionable insights from the analysis of agents. As part of an iterative development process, experts can assess the effect of changes across different designs, architectures, and even environment configurations. Based on analysis and comparison, experts should be able to derive clear steps for improvement.

4. Visual Analytics System: VISITOR

Based on the given user tasks, we designed ***VIS**ual **I**nteractive **S**tate Sequence Expl**O**ration for **R**einforcement learning (VISITOR)* as a visual analytics interface to visualize, explore and compare RL agents. Figure 3 shows the four main views embedded into a common user interface: (1) a state sequence embedding view **[E]**, (2) a reward view **[R]**, (3) a step detail **[A]**, and (4) a model list **[M]**. Each of the main views is applicable across the presented use cases and can be further customized to the specific requirements of the individual use case. Not visible in Figure 3, a sidebar contains controls of settings for the collection of state sequence data, as well as the settings for generating the embedding charts.

The main interactions are enabled by the linked state sequence embedding and reward views. In the reward view, the user can freely choose a time step via a slider according to patterns found in the reward function. Depending on the chosen scenario which we have described in the requirements, the views can be adapted. Task **T1**, analyzing the high-level behavior of an agent, can be seen as the default case and therefore is facilitated by the default configuration of the views (as seen in Figure 3). To support task **T2**, analyzing multiple policies, the model list provides small multiples, and coloring/highlighting to differentiate different policies is available. To accommodate user task **T3**, understanding of behavior, the environment detail view, and action probability view, paired with fine-grained time controls, allows a detailed diagnosis of decisions. Task **T4** is facilitated by the integration of VISITOR in an automated training and data generation framework, including experiment tracking that allows quick following-up on new results after design choices.

A major challenge directly emerging from the user tasks concerns the scalability of the application. An effective analysis of behavior over training might require looking at dozens of episodes, each of which might contain thousands of individual state-action transitions.

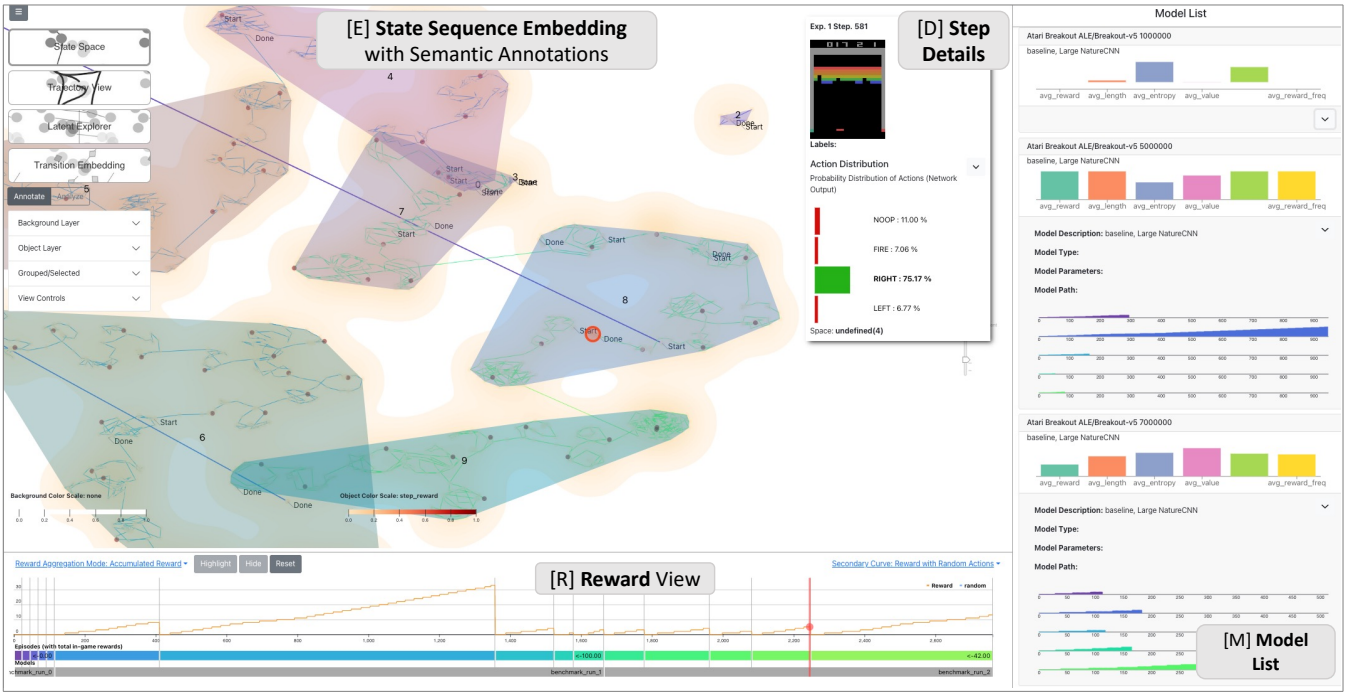

Figure 3: *The VISITOR interface applied to game-play sampled from a trained RL agent in the Atari game Breakout. The state sequence embedding view [**E**] shows a global overview of the state space and transitions. Convex hulls highlight connected clusters. Additional layers and annotations are available. The linked interactive reward view [**R**] captures the temporal structure of the sequences: the reward is plotted over steps. The step detail view [**D**] shows predicted action distribution and annotations. The model list [**M**] allows for easy model comparison.*

Existing tools and solutions using state space projections are not fully suited for full analysis because they lacked appropriate user interactions [ZZM16], were not designed to scale [WZY*21, JVW20], or lacked RL-specific filtering and information [HSH*21, BSH*15]. To facilitate the exploration and comparison of potentially large state-sequence sequences, our developed interface implements *overview* with *highlighting of important information*, *filtering and zooming* interactions, and *details on demand* [KAF*08] for state-action sequences in an RL context:

- Overview: The model list gives a metrics-based summary of the available models; at a high zoom level, single states are abstracted using density estimation or aggregation of trajectories. Users can select potentially interesting episodes based on patterns in the model list or by spotting value outliers in the aggregated state space embedding.
- Filtering and zooming: Sets of episodes can be selected and visually compared in a common embedding space. A large range of different values can be overlayed onto the state embedding view or reward view for detailed analysis. Four separate views show separate information, like state-activation mapping.
- Detail on demand: Single states and sequence fragments can be selected and investigated. Detail labels and images representing single states are revealed at appropriate zoom levels. Consecutive states are visually highlighted in relation to the selected states.

4.1. State Sequence Embedding View

To provide an intuitive and effective visualization of state sequences in reinforcement learning, we opted for a 2D embedding. Reinforcement learning often involves high-dimensional state spaces, making it difficult to visualize and directly analyze the relationships between states. By employing 2D embeddings, we can reduce the complexity of the state space and facilitate easier visualization and exploration of RL agents' behaviors and decisions. It allows users to easily comprehend and analyze the relationships between states without having to deal with the intricacies of high-dimensional data. We observed that UMAP [MHSG18] generally achieved superior results compared to PCA or t-SNE [vdMH08] because it struck a favorable balance between global and local expressiveness. Additionally, we optimize the embedding methods by computing globally consistent embeddings, i.e., all states are projected into a shared embedding and by using actions as additional information to encode state transitions via an action-angle optimized UMAP embedding. In the following, we briefly introduce the four views for the state sequence embedding:

State Space View Custom color scales can be applied to states, such as step reward, chosen action, and value prediction. An adaptive color scale displays the currently selected scale's meaning and value range. States are connected by trajectories highlighting the temporal flow of an episode within the embedding. Users can select a state by clicking, which influences the linked reward and step detail view. Users can also decide to show immediately proceeding or following states via a colored trace line. The length of the past and future traces can be adapted by the user. This makes it easy to follow a trajectory even with many overlying trajectories. States and trajectories can be dynamically filtered and highlighted, e.g., via a selection in the reward curve plots or a selection of single episodes, checkpoints, or models. The states inside the embedding

are automatically enriched via labels highlighting relevant states, such as episode starts. We found over-plotting to be problematic for large state spaces. We target this problem with a semantic zoom functionality: On a high zoom level, only density estimation plots are visible. When zooming in, additional details such as trajectories, states, and thumbnail images showing single states are revealed. As visible in the maximum zoom level, the step marker with past and future traces is displayed in Figure 4. Highlighted states stay permanently visible across zoom levels. As interpreting patterns or clusters in state space may be unclear without labels, we support progressive annotation and labeling, visualized in Figure 5. We use a spatiotemporal clustering algorithm (*ST-DBSCAN* [BK07]) to propose potential clusters that can be labeled by the user. The proposed clusters are visualized as convex hulls. By choosing a clustering algorithm that incorporates the temporal dimension, tightly connected sequentially areas of the state sequence space are well visible. The annotation is propagated to all underlying states and can be used in all other views. The expert can progressively label the state space, which improves knowledge generation. Figure 1a) shows this view with reduced visual options.

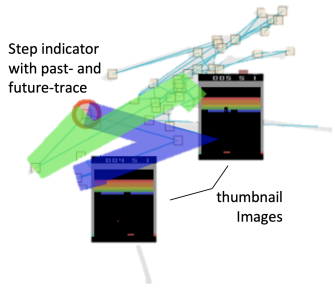

Figure 4: *Details at the maximum zoom level, including thumbnail images and the step indicator*

Decision Point View Compared to the state space view, the decision point view is designed to emphasize decision points inside the episodes, i.e., states at which the behavior of different policies diverges from each other. These points are often difficult to spot in state-focused embeddings, especially if only some overlapping trajectories reach the state. To enable the visual inspection of these points, the view focuses on visualizing the trajectories, with the states only visible on demand. We want policies that are aligned to merge visually, while the states at which the trajectories split should be highlighted. We use edge bundling by first merging states based on a defined minimum distance. Transitions between states that are connected to the same merged start and state are then merged. Compared to previous work, which, e.g., assumes chess with directly coinciding board positions [HSH*21, EHA*22], we allow continuous states and image observations which might be very similar but not equivalent. The state merging ensures that very similar states are treated as single states. We represent the size of the merged states by radius in the visualization. We encode the number of underlying trajectories by adapting the width of the line. The width of the line is normalized with the maximum amount of bundled lines. Decision points are states which represent a split of different trajectories: We identify these states by tracing the computing states in which the number of incoming and outgoing connections changes. Because similar episodes will be bundled into a common connection, we can assume that a change of connections indicates either a policy branching off or a policy merging with other policies again. The change of in- and outgoing connections is mapped onto a color scale. Figure 1b) shows the reduced visualization of the state space.

Mapping between Observations and Network Activations One goal of our workflow is to uncover the relationship between the observable behavior of an agent, and the internal representations. In the case of deep RL, these internal representations correspond to network activations. The tool implements a side-by-side plotting of observable states, equivalent to the previous views, and projection of the network activations. Then, states selected in the left observation embedding are visually connected to the corresponding states in the network activation embedding on the right side. With this view, analysts can uncover if an agent is not yet able to distinguish between different observable states because internal representations are not disentangled. We also enable a reverse-mapping by selecting a region in the right space. We have found that this view is insightful for analyzing multiple training checkpoints of the same model by revealing how the internal representations change over training and how they might be connected to specific predictions. Figure 1c) highlights how states and activations are visually connected.

Transition Embedding Finally, Figure 1d) shows the transition embedding view. Here, instead of computing and visualizing an embedding of states, we instead visualize an embedding of the state-to-state transitions, including the performed action. In particular, instead of embedding the states, we use the differences between states as a basis for two-dimensional embeddings. Furthermore, the action is appended to create a feature vector as input to the UMAP algorithm. To visually differentiate these from normal states, they are visualized as small diamonds colored according to the chosen action. Highlighted transitions are visually connected with both states, which are plotted in the background of the view. The view allows experts to identify related transitions across episodes.

4.2. Reward Graph & Detail View

The main curve of the reward view shows the reward for each continuous step of the sampled episodes. To better enable analysis, a user can switch between three modes: (1) The reward for each step, (2) the cumulative episode reward, and (3) the future accumulative reward in the episode, which corresponds to the true value. The secondary curve can be changed by the user to several custom metrics (connected to the available color scales in the embedding view). Users can, e.g., display the value prediction of the agent with the aforementioned ground-truth value, to spot discrepancies between predicted and real value. A discrepancy indicates an agent is not being fully trained on the affected parts of the state space. Because values for each step can be passed via the standardized info for each step (see [BCP*16] for details), a user-defined metric can be displayed in the graph, and accordingly chosen in the embedding state color scales. This is visualized in Figure 5. The user can interact with the reward plot to filter and highlight regions, which are propagated to the embedding view. The user can select value ranges, either to highlight via squared shapes instead of circles; or hide parts of the sequences. A time slider shows the correctly selected time step and lets the user jump to an arbitrary step in the sampled episodes. Below the reward chart, a compact hierarchy plot reveals the structure of the sampled episodes. Via a slider or control buttons, the user can switch the selected time step with varying levels of granularity. Linked to the other views, a **detail view** allows the fine-grained analysis of single steps. By selecting a step via the time slider or in

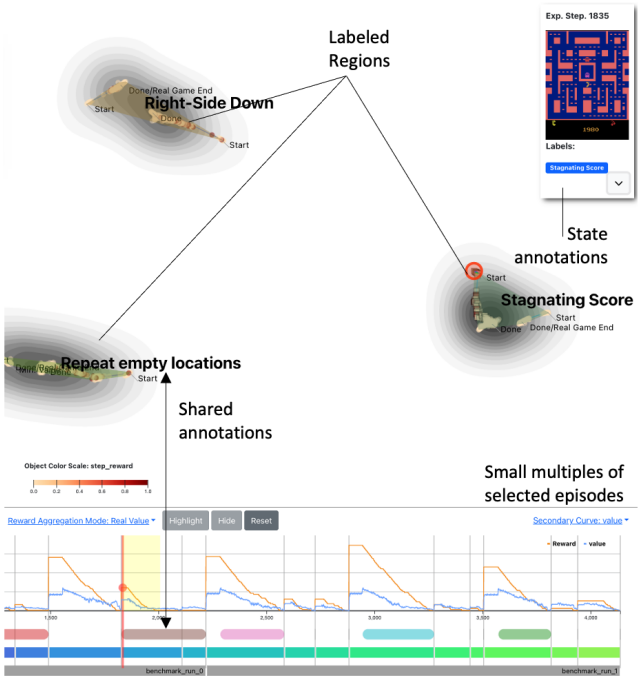

Figure 5: *Reward View and linked annotations.*

the embedding view, the environment details and action distribution are displayed. By default, a state rendering and annotations are displayed for the selected state. On-demand, additional details like the action distribution, raw input observations, and action distribution can be displayed. An environment rendering shows the internal state of the environment. Below the rendering, additional state-specific information like step reward, value prediction, steps until episode end, etc., is available. The raw observations passed to the agents can be visualized on demand. The raw input can deviate significantly from the rendering, e.g., if vector-valued observations are passed to an agent. The action probability diagram shows the distribution of predicted probabilities for actions at each state. The selected action is highlighted in green. This view allows the user to analyze the certainty of a predicted action, e.g., and to spot alternative decisions. For each state, an input attribution view visualizing input feature importance is available. The input attribution view is linked to the action probability view and shows the input attributions for the selected action.

4.3. Model List

As a last element of the approach, we introduce a model list, which gives an overview of the different benchmarked models. For each model, a separate card is rendered. Each card contains a summary glyph, which gives the most important statistical metrics like an average reward, episode length, or reward frequency (i.e., how often an agent achieves a reward). As a glyph, we choose a bar chart due to its familiarity with the involved experts. The size of the individual bars is normalized across the different models, which enables us to visually compare metrics for the different models. By hovering over a bar, the true value is displayed. Finally, users can expand the model cards to access additional information. This includes previously added model descriptions or tags. Further, an episode small multiple shows the accumulative reward for each episode sampled for a specific model. Highlighting states also affects the small multiples reward charts. The small multiples allowed experts to assess the performance across different episodes and create a visual mapping between the bottom reward chart and the single episodes. The model list is displayed in the right-side panel in Figure 3.

5. Evaluation: Expert User Study

We evaluate the use cases in a series of guided expert studies. Experts were selected based on their familiarity with machine learning in general and reinforcement learning specifically. We first conducted an expert study with six participants: Five of the surveyed experts (E1-E5) have multi-year practical and theoretical experience in reinforcement learning research with an academic background. Specifically, these five experts had experience with training and evaluating custom agents for different types of environments. This background allowed us to generate extremely insightful feedback based on hands-on experience. We have discussed both conceptual questions, as well as the actual implementation with the experts. One remaining expert (E6) has a more theoretical focus and less practical experience. While most experts were familiar with the scenario of Atari, they did not have detailed knowledge of the environment dynamics and agent strategies. A slight exception to this was the game of *Breakout*, in which the dynamic of tunnel-digging and eventual breakthrough was known to most participants based on anecdotal knowledge. The RL domain experts were generally not familiar with visual analytics but used experiment tracking tools like *Tensorboard*, *Weights&Biases*, as well as custom environment-specific visualizations.

Based on the first expert study feedback, we updated the tool, adding multiple new layers to the embedding view, as well as a model list. We report a detailed list of changes in the supplementary material. Most notably in the first study, experts did not use the embedding view to its full potential. We, therefore, focused on improving this aspect by adding additional layers and visual abstractions. A second shortcoming was the difficulty of comparing multiple agents head to head. We addressed this issue by introducing a model list view, which allows for quick metrics-based comparison and selection of models. We performed a second-phase expert study to evaluate the introduced changes. Specifically, two experts (E1, E2) of the first study, as well as two additional participants (E7, E8) reviewed the updated implementation. We asked the experts familiar with the tool to comment on changes made compared to the previous version. The additional participants also were active as research scientists in ML research, with E7 having a more theoretical background and E8 being very familiar with a broad range of areas in machine learning. With these experts, we went through the three use cases but also focused on feedback specific to the added visualizations.

5.1. Study Methodology

We evaluate the utility of the tool in three separate scenarios, using three Atari environments as a basis for analysis. Proceeding to the actual study, an initial free exploration phase enabled the domain experts to get familiar with the tool and explore its functionalities. While the participants were supported by a study leader, we chose minimal intervention in this initial phase to assess the intuitiveness and initial comprehensibility of the tool. Participants were not given a demo or video demonstration before this initial demonstration

phase. In a second experiment phase, three tasks were presented to each participant, with approximately 20 minutes for each task:

1. Analysis of a single agent in the game of *Breakout*. Participants were asked to first describe the general strategy and success of the agent. Then they were asked to identify success and failure events and come up with possible explanations for the identified events. They were also tasked with giving a suggestion for improvement (addressing tasks **T1**, **T4**).
2. Describing the evolution of a policy during training. As an example, the game *Seaquest* was chosen. Here, the participants were tasked to describe how the policy and the agent's behavior changed during training. In particular, the experts were asked to identify potential bottleneck skills, i.e., skills that are required to progress further in the game. (**T2**, **T3**)
3. Comparison of agents in the game of *MsPacman* (a variant of the popular original Pacman arcade game). Here, the experts were asked to compare the characteristics of two different agents trained in different training configurations. The comparison was requested both in terms of performance and behavior. Finally, the experts were asked which agent (and in extension training configuration) to prefer, and to reason about the effect of parameter settings. (**T2**, **T3**, **T4**).

For each of the three tasks, interference was limited to the suggestion of tool functionalities, and the review of intermediate results. After solving the tasks via self-developed strategies, we explicitly asked participants to comment on previously unused functionalities and visual elements. This approach allowed us to observe the participant's learning curve and independent problem-solving with relatively little bias, while still requesting opinions on all major elements of the application.

5.2. Expert Case Study Results

In the following, we give brief descriptions of the three use cases.

5.2.1. Evaluation of a Single Policy

We chose the *Breakout* environment for our initial use case due to its simplicity and familiarity, as illustrated in Figure 3. The experts were tasked with analyzing an agent trained using a policy-gradient algorithm (PPO [SWD*17]) to sub-optimal performance.

The initial study revealed the effectiveness of combining the linked temporal reward view, individual state renderings, and the embedding view for exploring state sequences. The reward curve served as a valuable anchor, enabling experts to easily identify interesting states, differentiate episodes, and navigate within episodes. By examining the reward view alongside visual inspection of single states, experts quickly recognized the sub-optimal performance of the presented agent. They identified a "breakthrough" event, and linked it to the state sequence embedding space. Overall, the experts rated the approach's suitability for the first use case favorably, with an average score of 6/7 (Std.Dev. 0.57).

However, the state sequence embedding views were often underutilized as they were challenging to interpret and sometimes overwhelming. After a detailed explanation of the methodology, most experts were able to partially understand the relationship between in-game states and positions in the state embedding. We observed that we needed to encourage participants to fully utilize the embedding, for example, by highlighting the possibility to change color scales, zooming, or filtering in the reward view.

To address this concern, we introduced additional view options, which improved the overall understanding of the embeddings and provided clearer distinctions between observation and latent representations. In particular, the decision point view was well received for its visual simplification of the state sequence space, as expressed by E7: "[the] decision point view is actually my favorite." The four participants of the second expert study rated it highly, with an average score of 5.5/7 (Std.Dev. 1.1).

5.2.2. Describing the Evolution of a Policy over Training

In the second use case, we selected the *Seaquest* game as the environment, where the agent must shoot enemy entities to collect points while managing limited oxygen storage. The game consisted of 5 episodes each for 5 checkpoints: the initial random agent, the fully trained agent, and 3 intermediate checkpoints. We performed this use case with 5 out of the 8 experts due to technical difficulties causing delayed user input.

By analyzing the reward chart, experts identified a positive training progression, with the random agent achieving short episodes and small rewards, and the fully trained agent showing much better results. All experts independently identified a breakthrough in training, which led to significantly higher episode rewards. They confirmed through visual inspection that the agent learned to refill oxygen at the surface, thereby prolonging the game.

Two experts first identified this behavior via the state sequence embedding view, which displayed a distinctive pattern and emergence of clusters in the global state space. They inferred the agent's general strategy and observed that the y-position in the embedding space roughly corresponded to the agent's position in the environment. The random agent's states appeared as a separate cluster, as it mostly acted in the bottom part of the screen. The x coordinate in the state sequence space strongly correlated with the value prediction of the state, leading to a characteristic pattern for the refilling behavior. The value function coincided with the oxygen level, acting as a strong indicator of the remaining episode length and value estimation. The 5 participants in this use case evaluation rated the application's general utility highly, with an average score of 6/7 (Std. Dev. 0.63).

The introduced model list provided additional benefits, enabling easy comparison between model checkpoints based on key metrics like rewards, informing model selection and subsequent analysis. Experts noted that reward stagnated during training after an initial improvement, attributable to the agent's inability to overcome the specified behavior of collecting divers before refueling.

5.2.3. Comparison of Two Policies

In the final use case, we selected the *MsPacman* game to test the user's ability to directly compare two trained policies with slightly different training configurations. The state sequence space for this game appeared less contiguous and more fragmented into smaller clusters, each showing homogeneity in various properties. Experts identified a re-emerging pattern in the reward view and observed that both agents could get stuck at different locations. These "stuck

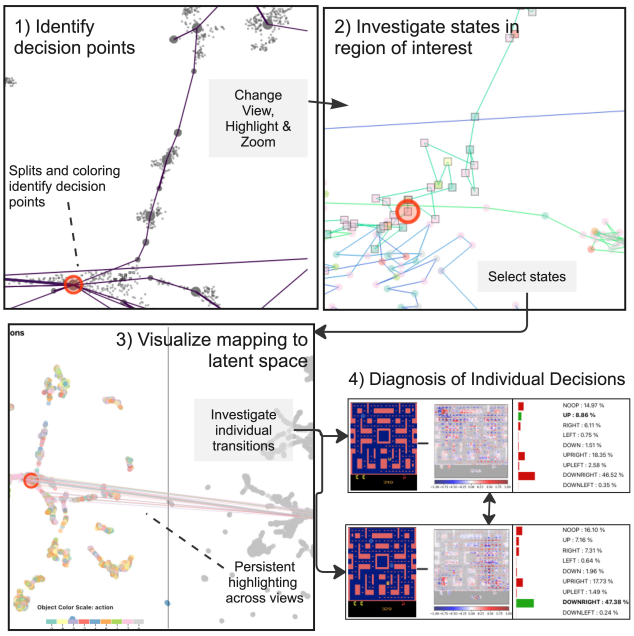

Figure 6: *The figure shows an exemplary analysis workflow during the comparison of two models: First, a state of interest is highlighted in the decision point view. The state is annotated and highlighted, to be further investigated. Different views, e.g., reveal how different models diverge at this point and show that states are indeed mapped to different locations in the latent space. The difference in probability distributions also shows the different preferences.*

states" were evident in the state embedding space, especially when using the *action* color scale. Most experts recognized that agents got stuck at consistent points across different runs. Experts generated hypotheses for agent behavior, with one concluding that agent 1 got stuck in a corner due to the absence of an immediate reward. During the comparison, agents could split the embedding view window, allowing them to focus on different regions of the state space, including separate controls and color scales.

The initial expert study rated the tool's suitability for comparison lower than for other use cases, with a score of 4.4/7 (Std.Dev. 0.63). Experts suggested improvements such as side-by-side state comparisons and synchronous replay of environment detail views. In response, we increased the visual differentiation between models in the reward view and introduced new views like decision point and activation mapping views to support a more comprehensive analysis of multiple policies, as shown in Figure 6.
These improvements led to a notable increase in the tool's suitability rating for comparison, with a score of 6.0/7 (Std.Dev. 0.0) from two new experts. Second-time participants also provided positive feedback on the enhanced workflow for comparing agents.

5.3. Qualitative Feedback

General expert feedback on the application was generally positive, despite some smaller technical issues occurring. The experts shared some ideas for further improvement, which were partly already integrated into the final version of the implementation: These suggestions include additional color scales, like steps until the end of an episode (E2), a better visual distinction between different episodes, and models in the reward view (E3, E4) increasing the size of the step selection slider in the reward view (E3).
An important aspect of the usability of the tool is the time experts need to explore and learn the tool's functionalities. Here, the comparatively low number of linked views was advantageous because experts were able to explore all main components of the interface in a short amount of time. Furthermore, we chose a few nested or hidden interfaces which contributed to the tool being perceived as compact. In earlier iterations of the prototype, instead of providing explicit controls of e.g., the content displayed in the state sequence embeddings, we more heavily relied on indirect ways to determine the displayed information, in particular via semantic zoom. However, we found providing a selected set of explicit and meaningfully named controls helped the expert users to better interpret the semantics of the shown visualizations, in particular the state sequence embedding. Looking at the state-sequence embedding, experts sometimes failed to make sense of the displayed visual representations. Giving clear options such as *State Space* or *Decision Points*, further emphasized by sketched visualizations helped experts to better interpret the shown plots. In general, we also added multiple labels or descriptions and improved the onboarding with an introductory modal at application start-up, and a dedicated explanation of the embeddings.

All experts, particularly those who train agents regularly, approved the potential usefulness of the tool after the study and could imagine using the tool for certain use cases in the future once available. Some experts worried about the tool being difficult and lengthy to set up and the need to save and collect episode data. These experts were thus pleased with the existing integration into common frameworks and automated data generation. The potential ease of integration into the existing workflow, as well as the tool being agnostic to algorithm and environment, but also extendable, was well received.

5.4. Quantitative Feedback

We found VISITOR to be effective in supporting users in the presented use cases: In the initial user study, we found all experts to effectively analyze and describe learned agent behavior. In a post-experiment survey, the experts rated ease of use and effectiveness as high (5.2/7, Std.Dev. 0.68), completeness as very high (5.6/7, Std.Dev. 0.45), ease of use generally high (4.9/7, Std.Dev. 0.98), and reported generally low levels of frustration except for limited technical errors (2.6/7 with 1 meaning low frustration). The interaction between environment rendering, reward chart, and time slider, which quickly felt familiar to all experts with experience in training RL models, ensured basic utility across all tasks and scenarios. State Sequence embeddings, detail view, and action probability view added additional abilities to the analytical capabilities of agents. The additional design elements were well received by the participants of the second user study. The additional views were rated highly, with both second-time participants confirming a notable improvement in the application. For **T1**, 8/8 experts could identify critical events like the "breakthrough" or episode/game ends. For **T2**, 4/5 experts surveyed for the second task were able to not only spot the behavior of the initial untrained agent as an outlier but also found the critical skill of "refilling oxygen" which was only available to agents progressed further in the training. We evaluate **T3** multi-faceted as it is harder to de-

fine: In the **Breakout** environment, 6/8 experts managed to spot narrow misses of the paddle, i.e., the expert missing shots it should have been able to catch by performing an imprecise action just before impact. In the *Seaquest* environment, only 2/5 experts were able to spot the collection of divers as a necessary element enabling the game mechanic of refilling. Finally, in the *MsPacman* use case, 6/6 experts correctly identified the stagnation of collected reward and were able to identify the underlying cause of the behavior. Finally, experts also were able to address use case **T4**. In particular, 6/8 experts correctly identified a change in sampling strategy (from sampling the distribution to just choosing the action with the highest probability) as an effective counter-strategy, preventing some narrow misses.

6. Discussion

VISITOR is designed to be agnostic to the domain, the environment, or the algorithm. This greatly improves its applicability to a wide range of different scenarios, both in research and in real-world applications. Therefore, evaluating the tool in varying application contexts beyond standard benchmark environments would be very interesting. The current tool is mainly targeted and was evaluated by RL experts. For future applicability, it is interesting if domain experts, e.g., engineers that are only familiar with basic RL concepts, can generate value for the developed application. RL, as a whole, is not yet at the level of commodification that other types of models, such as image classification or language models are. As VISITOR is highly modular, one could imagine extending it towards particular tasks, as we have seen in past work, e.g., heatmaps for spatial navigation tasks [JVW20, HLB*20], or the custom interactive rendering of the environment state, etc.

We already applied our application in one exemplary medical domain, visualizing medical device trajectories during simulated operations. This application showcases our approach's scalability. Although conceptually simple, the chosen games exhibit challenges like high-dimensional image-based observation spaces, episodes spanning thousands of steps, and varying behavior across models. Single episodes span thousands of steps with complex behavior in three-dimensional space. Figure 7 displays the joint embedding of two trajectories by agents with varying skill levels. Aligned trajectories represent the same procedure, with distinguishable clusters corresponding to sub-procedures. Inexperienced agents need more time and move the instrument to a larger degree, which is clearly identifiable in the embeddings. Future iterations plan to visualize simulated behavior and agent learning via reinforcement learning. While certain scalability limitations remain, in particular when the computed embeddings are not able to produce embeddings that allow for a differentiation between different states and clusters, it highlights the flexibility of our approach.

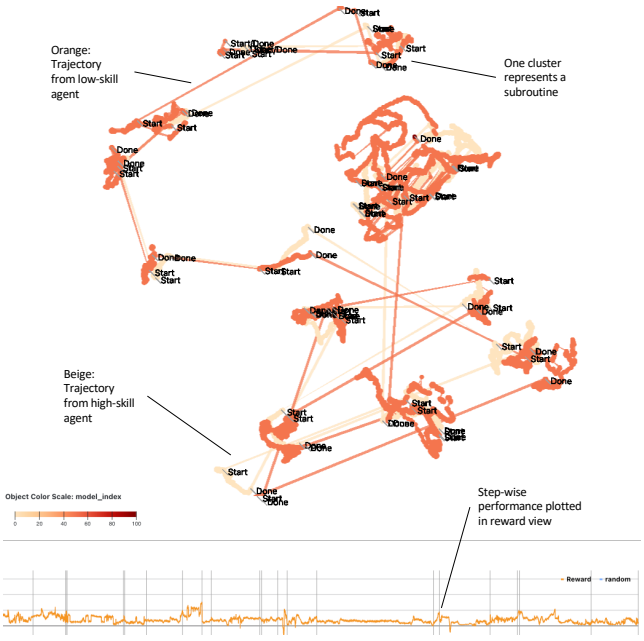

Figure 7: *The figure shows two trajectories in a complex real-world use case loaded into VISITOR. Orange and beige each correspond to a separate state sequence produced by two separate agents.*

Similar to Figure 7, our approach suits itself to, e.g., compare learned behavior with expert trajectory data and is therefore well applicable to imitation learning. Furthermore, plotting the full state space of a training RL agent could be used in active learning scenarios, e.g., by a human pointing out under-explored parts of the state space. However, by default, the embeddings cannot show regions of the state space that are yet unexplored, i.e., do not have any data points. We like that note that the tool is also applicable in scenarios with meaningful 2D coordinates, e.g., in a navigation problem [JVW20]. Here, instead of 2D coordinates from a projection, we can directly use 2D coordinates from an environment. VISITOR is applicable to such a use case without modifications.

Finally, VISITOR still operates on the scale of a relatively small set of checkpoints or models and is targeted at development as well as potential verification and presentation to end-users. Past work has already explored workflows targeted at larger sets of different models/agents during training (e.g., Saldanha et al. [SPBA19]). While the model list is a first step towards this goal, extending the interactions presented in VISITOR with capabilities to compare and browse even larger sets of different agents and model configurations could further improve experimentation and development.

7. Conclusion

We presented VISITOR, a versatile, general application for the interactive analysis of sequences generated by RL agents. The application provides linked views like a 2D state sequence embedding, a model list and a temporal reward view. The state sequence embeddings are enriched with abstractions and annotations to support the exploration of complex spaces. We showcase how the tool can be applied to analyzing single policies, understanding the evolution of a policy during training, and comparing agents. The implementation is well-received among RL experts. The tool is available at: https://visitor.dbvis.de.

Acknowledgement

This work was supported by the German Research Foundation as part of the priority programme "Volunteered Geographic Information: Interpretation, Visualisation and Social Computing" (VGIscience, priority programme 1894). Funding was partially provided

by ETH Zurich. Open Access funding enabled and organized by Projekt DEAL.

References

[AB18] ADADI A., BERRADA M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access 6* (2018), 52138–52160. doi:10.1109/ACCESS.2018.2870052. 1

[ADBB17] ARULKUMARAN K., DEISENROTH M., BRUNDAGE M., BHARATH A.: A brief survey of deep reinforcement learning. *IEEE Signal Processing Magazine 34* (08 2017). doi:10.1109/MSP.2017.2743240. 1

[AS19] ANNASAMY R. M., SYCARA K.: Towards better interpretability in deep q-networks. *Proc. of the AAAI Conf. on Artificial Intelligence 33* (7 2019), 4561–4569. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4377, doi:10.1609/AAAI.V33I01.33014561. 3

[BCP*16] BROCKMAN G., CHEUNG V., PETTERSSON L., SCHNEIDER J., SCHULMAN J., TANG J., ZAREMBA W.: Openai gym, 2016. arXiv:arXiv:1606.01540. 6

[BK07] BIRANT D., KUT A.: St-dbscan: An algorithm for clustering spatial–temporal data. *Data and Knowledge Engineering 60*, 1 (2007), 208–221. Intelligent Data Mining. URL: https://www.sciencedirect.com/science/article/pii/S0169023X06000218, doi:https://doi.org/10.1016/j.datak.2006.01.013. 6

[BNVB13] BELLEMARE M. G., NADDAF Y., VENESS J., BOWLING M.: The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research 47* (jun 2013), 253–279. doi:10.1613/jair.3912. 3

[BSH*15] BACH B., SHI C., HEULOT N., MADHYASTHA T., GRABOWSKI T., DRAGICEVIC P.: Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics PP*, 99 (2015), 1–1. doi:10.1109/TVCG.2015.2467851. 3, 5

[DLM*20] DULAC-ARNOLD G., LEVINE N., MANKOWITZ D. J., LI J., PADURARU C., GOWAL S., HESTER T.: An empirical investigation of the challenges of real-world reinforcement learning. *arXiv e-prints* (Mar. 2020), arXiv:2003.11881. arXiv:2003.11881, doi:10.48550/arXiv.2003.11881. 1

[EHA*22] ECKELT K., HINTERREITER A., ADELBERGER P., WALCHSHOFER C., DHANOA V., HUMER C., HECKMANN M., STEINPARZ C., STREIT M.: Visual exploration of relationships and structure in low-dimensional embeddings. *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–1. doi:10.1109/TVCG.2022.3156760. 1, 3, 6

[GKDF18] GREYDANUS S., KOUL A., DODGE J., FERN A.: Visualizing and understanding Atari agents. vol. 80 of *Proc. of Machine Learning Research*, PMLR, pp. 1792–1801. URL: http://proceedings.mlr.press/v80/greydanus18a.html. 2, 3

[HAB*20] HRISTOV Y., ANGELOV D., BURKE M., LASCARIDES A., RAMAMOORTHY S.: Disentangled relational representations for explaining and learning from demonstration. In *Conference on Robot Learning* (2020), PMLR, pp. 870–884. 3

[HCDR21] HEUILLET A., COUTHOUIS F., DÍAZ-RODRÍGUEZ N.: Explainability in deep reinforcement learning. *Knowledge-Based Systems 214* (2021), 106685. URL: https://www.sciencedirect.com/science/article/pii/S0950705120308145, doi:https://doi.org/10.1016/j.knosys.2020.106685. 3

[HIB*] HENDERSON P., ISLAM R., BACHMAN P., PINEAU J., PRECUP D., MEGER D.: Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18, AAAI Press. 1

[HLB*20] HE W., LEE T. Y., BAAR J. V., WITTENBURG K., SHEN H. W.: Dynamics explorer: Visual analytics for robot control tasks involving dynamics and lstm-based control policies. *IEEE Pacific Visualization Symposium 2020-June* (6 2020), 36–45. doi:10.1109/PACIFICVIS48177.2020.7127. 1, 2, 3, 10

[HS97] HOCHREITER S., SCHMIDHUBER J.: Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735—-1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735. 3

[HS15] HAUSKNECHT M. J., STONE P.: Deep recurrent q-learning for partially observable mdps. *CoRR abs/1507.06527* (2015). URL: http://arxiv.org/abs/1507.06527, arXiv:1507.06527. 3

[HSH*21] HINTERREITER A., STEINPARZ C. A., HECKMANN M., STITZ H., STREIT M.: Projection path explorer: Exploring visual patterns in projected decision-making paths. *ACM Transactions on Interactive Intelligent Systems 11*, 3–4 (2021), Article 22. URL: https://dl.acm.org/doi/10.1145/3387165, doi:10.1145/3387165. 2, 3, 5, 6

[Irp18] IRPAN A.: Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018. 1

[JVW20] JAUNET T., VUILLEMOT R., WOLF C.: Drlviz: Understanding decisions and memory in deep reinforcement learning. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 49–61. 3, 5, 10

[KAF*08] KEIM D., ANDRIENKO G., FEKETE J.-D., GORG C., KOHLHAMMER J., MELANÇON G.: Visual analytics: Definition, process, and challenges. *Lecture notes in computer science 4950* (2008), 154–176. 5

[LSSP21] LIU G., SUN X., SCHULTE O., POUPART P.: Learning tree interpretation from object representation for deep reinforcement learning. *Advances in Neural Information Processing Systems 34* (12 2021). 3

[MBP*23] MOERLAND T. M., BROEKENS J., PLAAT A., JONKER C. M., ET AL.: Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning 16*, 1 (2023), 1–118. 1

[MHSG18] MCINNES L., HEALY J., SAUL N., GROSSBERGER L.: Umap: Uniform manifold approximation and projection. *Journal of Open Source Software 3*, 29 (2018), 861. URL: https://doi.org/10.21105/joss.00861, doi:10.21105/joss.00861. 5

[MKS*13] MNIH V., KAVUKCUOGLU K., SILVER D., GRAVES A., ANTONOGLOU I., WIERSTRA D., RIEDMILLER M. A.: Playing atari with deep reinforcement learning. *CoRR abs/1312.5602* (2013). URL: http://arxiv.org/abs/1312.5602, arXiv:1312.5602. 1, 2

[MMSV20] MADUMAL P., MILLER T., SONENBERG L., VETERE F.: Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence* (2020), vol. 34, pp. 2493–2500. 3

[MSHB22] MISHRA A., SONI U., HUANG J., BRYAN C.: Why? why not? when? visual explanations of agent behaviour in reinforcement learning. In *2022 IEEE 15th Pacific Visualization Symposium (PacificVis)* (Los Alamitos, CA, USA, apr 2022), IEEE Computer Society, pp. 111–120. URL: https://doi.ieeecomputersociety.org/10.1109/PacificVis53943.2022.00020, doi:10.1109/PacificVis53943.2022.00020. 1, 2

[MSS*] METZ Y., SCHLEGEL U., SEEBACHER D., EL-ASSADY M., KEIM D. A.: A comprehensive workflow for effective imitation and reinforcement learning with visual analytics. In *13th International EuroVis Workshop on Visual Analytics (EuroVA 2022)*, pp. 19–23. 1

[NIAN] NIKULIN D., IANINA A., ALIEV V., NIKOLENKO S.: Free-lunch saliency via attention in atari agents. *Proc. of Intl. Conf. on Computer Vision Workshop, ICCVW 2019*, 4240–4249. doi:10.1109/ICCVW.2019.00522. 2, 3

[PCC*] PAN X., CHEN X., CAI Q., CANNY J., YU F.: Semantic predictive control for explainable and efficient policy learning. In *Proc.*

of IEEE Intl. Conf. on Robotics and Automation, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., pp. 3203–3209. doi:10.1109/ICRA.2019.8794437. 3

[SB18] SUTTON R. S., BARTO A. G.: *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. 2

[SG20] SEQUEIRA P., GERVASIO M.: Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence 288* (2020), 103367. 3

[SHS*17] SILVER D., HUBERT T., SCHRITTWIESER J., ANTONOGLOU I., LAI M., GUEZ A., LANCTOT M., SIFRE L., KUMARAN D., GRAEPEL T., LILLICRAP T. P., SIMONYAN K., HASSABIS D.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR abs/1712.01815* (2017). URL: http://arxiv.org/abs/1712.01815, arXiv:1712.01815. 1

[SHS*22] SHI W., HUANG G., SONG S., WANG Z., LIN T., WU C.: Self-supervised discovering of interpretable features for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 5 (2022), 2712–2724. doi:10.1109/TPAMI.2020.3037898. 3

[SHSW21] SHI W., HUANG G., SONG S., WU C.: Temporal-spatial causal interpretations for vision-based reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (12 2021). doi:10.1109/TPAMI.2021.3133717. 3

[SMG22] SINHA S., MANDLEKAR A., GARG A.: S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Proc. of Conf. on Robot Learning* (08–11 Nov 2022), Faust A., Hsu D., Neumann G., (Eds.), vol. 164 of *Proc. of Machine Learning Research*, PMLR, pp. 907–917. URL: https://proceedings.mlr.press/v164/sinha22a.html. 1

[SPBA19] SALDANHA E., PRAGGASTIS B., BILLOW T., ARENDT D.: ReLVis : Visual Analytics for Situational Awareness During Reinforcement Learning Experimentation. In *EuroVis (Short Papers)* (2019), Eurographics Association, pp. 43–47. doi:10.2312/evs.20191168. 3, 10

[SVZ13] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013). 1

[SWD*17] SCHULMAN J., WOLSKI F., DHARIWAL P., RADFORD A., KLIMOV O.: Proximal policy optimization algorithms. *CoRR abs/1707.06347* (2017). URL: http://arxiv.org/abs/1707.06347, arXiv:1707.06347. 8

[Ten] TENSORFLOW: Tensorflow/tensorboard: Tensorflow's visualization toolkit. URL: https://github.com/tensorflow/tensorboard. 4

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-sne. *Journal of Machine Learning Research 9*, 86 (2008), 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html. 5

[VMS*18] VERMA A., MURALI V., SINGH R., KOHLI P., CHAUDHURI S.: Programmatically interpretable reinforcement learning. *35th International Conf. on Machine Learning, ICML 2018 11* (2018), 8024–8033. 3

[WGSY19] WANG J., GOU L., SHEN H. W., YANG H.: Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE Transactions on Visualization and Computer Graphics 25* (2019), 288–298. doi:10.1109/TVCG.2018.2864504. 1, 2, 3

[WZY*21] WANG J., ZHANG W., YANG H., YEH C. C. M., WANG L.: Visual analytics for rnn-based deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics* (2021). doi:10.1109/TVCG.2021.3076749. 2, 3, 5

[ZZM16] ZAHAVY T., ZRIHEM N. B., MANNOR S.: Graying the black box: Understanding dqns. In *Proc. of Intl. Conf. on on Machine Learning - Volume 48* (2016), ICML'16, JMLR.org, p. 1899–1908. 2, 3, 5