EUROGRAPHICS 2023 / D. Archambault, R. Bujack, and T. Schreck
(Guest Editors)

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

Supplementary Material for 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

Abstract

Understanding the behavior of deep reinforcement learning (RL) agents is a crucial requirement throughout their development and deployment. 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. The system specifically addresses the requirements of reinforcement learning experts by enabling a domain-agnostic progressive analysis and annotation workflow. 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. This view contains semantically enriched state sequences based on generated labels and manual annotation. 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. An automated guided tour can be generated based on the user-generated annotations. Our evaluation with six RL experts confirms the effectiveness in identifying states of interest, comparing the quality of policies, and reasoning about the internal decision-making processes. The tool supports experts for debugging of models, understanding of their limitations, and provides a way to communicate complex properties of RL agents to other users.

CCS Concepts

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

1. Extended Related Work: Explainable AI for Reinforcement Learning

Due to space constraints, we have omitted some additional related research that had a direct impact on our work. Beyond the research area of visual analytics, there is additional research, in particular in explainable AI, which also inspired part of this work.

Projection-Based Explanations – In a first contribution, Zahavi et al. [ZZM16] utilize a t-SNE [vdMH08] projection of the neural activations of states encountered by agents. The clustering of states is performed based on hand-crafted features, e.g., the position of in-game elements of Atari games. Additionally, this approach uses a skill-discovery algorithm to highlight state transitions between high-level clusters. Compared to their approach, we utilize projections that preserve the global structure, facilitating the tracing of state sequences. This allows, e.g., the comparative analytics between multiple agents. Furthermore, our approach requires no manual feature extraction but instead allows simple annotations in the tool.

Attribution-Based Explanations – Greydanus et al. [GKDF18] and Shie et al. [SHS*20] propose methods that probe the effect of input perturbations on action selections, i.e., they determine which input regions have the strongest effect on decision-making, generally called attribution methods. These techniques are related to existing interpretability methods for supervised learning, which use masking or perturbations and probe their effect on model output, such as LIME [RSG16]. Attribution maps are often integrated into existing frameworks due to their simplicity, both for implementation and interpretation by users [ZZM16, WGSY19]. However, the reliability of these methods is often unclear [AGM*18], and multiple interviewed experts raised concerns about their use. We, therefore, decided to use them at selected steps.

Sample-Based Explanations – Sequeira et al. [SG20] present a general framework to extract steps in the training/testing phase based on a conceived interestingness measure, with the goal to explain key behaviors in a condensed form. We draw on the basic idea by integrating measures of interestingness and leading users to potential behaviors based on both automated analysis and visual inspection.

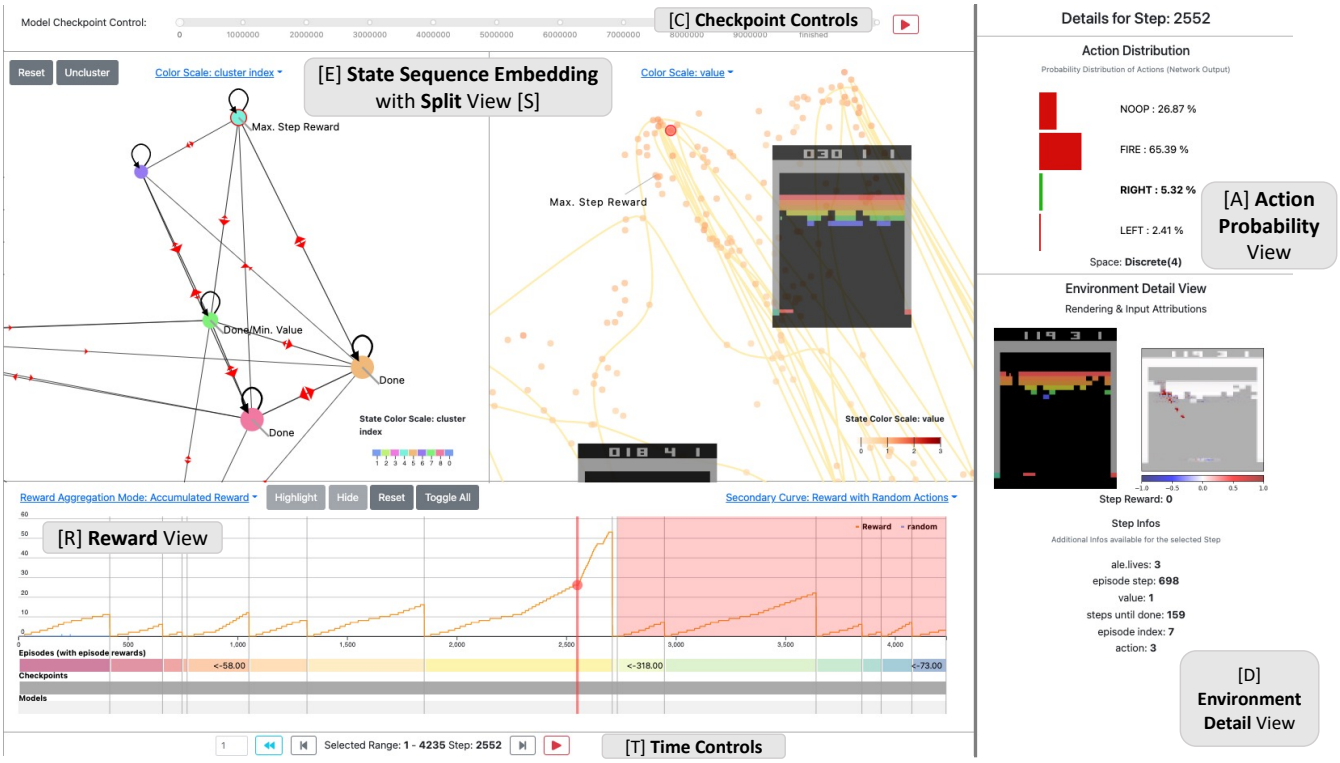


Figure 1: *The previous iteration of the main interface design. In the state-sequence embedding view **[E]**, comparison of models was enabled by a split view **S**. In the new version, we opted to visualize all trajectories in the same view, but improved filtering, abstraction, and visual differentiation. Compared to the version presented in the main paper, the model list was missing. Instead, space was taken up by the detail view, which is no more compact in the updated design. The previous version did not provide comparable functionality in terms of state-sequence abstractions and annotations, which limited the utility of the state-action sequence embedding view.*

2. Previous Design Iterations

Fig. 1 shows a previous design iteration of the tool. While core elements of the application were present, e.g., the linked state sequence embedding and temporal reward views, there were several omissions: Connected state sequences were used, however, visual abstractions were more limited: While hierarchical clustering was implemented, it required manual user input to choose the number of clusters, and analysts did not find it fully useful. We decided to build on this by creating four views on the same underlying data which we presented in the main paper. Furthermore, we added or changed the following additional designs:

1. Right Pannel: In the old version, the right panel of the application was occupied by an action probability view and an environment detail view, showing summary statistics. While this was relevant information, used extensively by the experts, it took up a large amount of screen space. In the new version, we significantly miniaturized the detail view by putting the rendering in the top-right corner of the sequence embedding view, with details on demand.
2. The comparison of different models was a major goal for the application. However, in the previous version, agent comparison was not supported sufficiently. To overcome this issue, we added a model list in the right panel of the application. The model list allows for a quick comparison between models based on important metrics such as average reward, episode length, or action entropy, i.e., how much an agent explores.
3. We merged the checkpoint controls with the model list, simplifying the user interface.
4. We added multiple ways to annotate the state space with persistent labels, which enables saving, presenting, and exporting the generated knowledge.

3. Details on the embedding and 2D projections

Ensuring suitable 2D projection is crucial as it is the centerpiece of the application. Therefore, as mentioned in the main paper, considerable effort was necessary to ensure that the projection algorithms were able to process the state sequence data. Here, we want to give important implementation details, which aim to support future work.

3.1. Optimizing Unsupervised State Sequence Embedding

To represent trajectory and policy space in a general and scalable way, we rely on embeddings. We have experimented with different methods to further optimize the state sequence embedding to RL specifics.

Basic Embeddings – We choose *UMAP* [MHM18] as the basis for a dimensionality-reduction algorithm. We investigated both raw

observations, i.e., the input to the neural network, and the output of the last layer of the used neural feature extractor to compute two-dimensional embeddings of high-dimensional input observations. Feature extractors are part of deep learning architectures to encode, e.g., images, to a policy. The latent feature vector does not have to be a specific shape or structure, as the computation of embedding relies only on similarity.

Action-Angle Loss – As an additional element contributing to the state sequence embeddings, we experimented with encoding actions in a globally consistent way. The performed action that leads from one state to the next one is encoded as the angle between the two sequential states. The action-angle loss is defined either as a categorical action distribution or a one-dimensional continuous action. An action is mapped onto a distinct angle, e.g., 4 available actions would be mapped onto $[0°, 90°, 180°, 270°]$. An auxiliary loss is added to the cost function of the final embedding training step of *ParametricUMAP*. The original cost function for *ParametricUMAP* by Sainsburg et al. [SMG20]:

$$C_{UMAP} = \sum_{i \neq j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) + (1 - p_{ij}) \log\left(\frac{1 - p_{ij}}{1 - q_{ij}}\right)$$

Hence, the adapted cost function incorporating the action-angle loss is:

$$C_{UMAP+Angle} = C_{UMAP} + \beta \left\| \hat{\theta}_{a_t} - \theta_{s_{t+1},s_t} \right\|$$

with $\theta_{s_{t+1},s_t} = \text{atan2}(y_{s_{t+1}} - y_{s_t}, x_{s_{t+1}} - x_{s_t})$. Here x and y represent the positions of the respective states in the embedding space. $\left\| \hat{\theta}_{a_t} - \theta_{s_{t+1},s_t} \right\|$ represents the real distance between angles, i.e., an angle of 0.05 and $2\pi - 0.05$ is very similar. A weight factor β determines the trade-off between the original optimization objective and the action-angle loss. We have generally achieved good results with a β-weight of 0.1. We have seen, that applying the action-angle auxiliary loss leads to very consistent embedding between different runs, in particular when the distribution of actions is similar. On the flip side, when only a small number of actions is used, the embedding can get skewed in a particular direction. We plan to further develop and evaluate UMAP embeddings with auxiliary action-angle losses as a general approach for the visualization of state-action sequences in future work.

Including Temporal Information – A second way of introducing RL-specific information is to concatenate the episode step onto the embedding input vector. In Figure 2, we present the effect of this episode step concatenation with different weighting factors. Besides the implicit temporal weighting, our tool also allows plotting a one-dimensional UMAP embedding, with the time step being on the vertical axis.

Stable Embeddings for Model Comparison – As described, one main development goal was to enable comparison between agents and/or checkpoints. E.g., we want the sequence embedding to be stable over training time to show the change in visited state distribution, i.e., we want the same embedding position for a visited state at any time during training. The same goes for two or more different models. This is achieved by using a common, fixed embedding function for any possible state. For low-dimensional observations (e.g., kinematics vectors, etc.), we can rely on the raw features as input to UMAP. For high-dimensional observations like images, we apply a re-projection of observations via the feature extractor of the best available model (which should provide wide coverage of the state space) to extract the latent features for ParametricUMAP. While the usage of a single neural feature extractor introduces a bias in terms of similarity, we assume the latent features to be informative in terms of embedding position. Furthermore, the trained feature extractor is already available as a result of the RL training. To achieve a non-biased embedding of high-dimensional features, we may use an auto-encoder-style neural network trained from scratch. To still faithfully represent model-specific internal representations, we found the split view very effective: Displaying, e.g., the common projection of observations on the left, and the individual projections of latent features on the right. We often can identify meaningful dimensions in projections of latent features. In particular, we found that the value of a state is often implicitly modeled by the neural network, and is therefore encoded in the latent features. We highlight this observation in Figure 3.

4. Implementation Details

We implement VISITOR as a stand-alone web application based on *React* and *d3.js*. To support the rendering of large state spaces, we used the *regl-scatterplot* [Lek] library, with the default render and additional elements being implemented with *d3* and *canvas*. We tightly integrate the application with open-source frameworks for reinforcement learning (*StableBaselines3* [RHE*19]) and imitation learning (*Imitation* [WTGE20]), widely used throughout the RL research community. Furthermore, more generic experiment tracking tools like *Tensorboard* are also integrated as separate tabs. At any point, recorded episodes, as well as computed intermediate values and embeddings can be saved and reused for future use. In particular, the annotated state sequences can be saved and reloaded, which enables sharing results, including annotations.

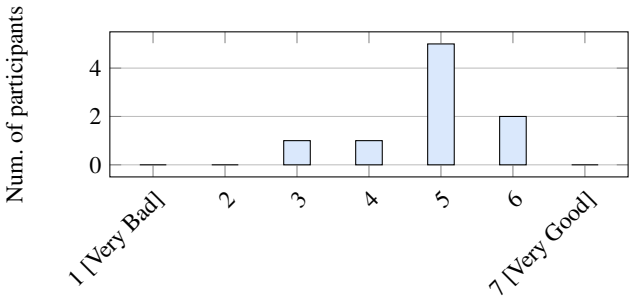
To fully support iterative model development, including retraining of models, VISITOR is integrated into a multi-page web application that supports the entire development and evaluation workflow (available as open-source).

5. Detailed Expert Study Results

In the following, we want to give some additional details of the expert study, including the full distribution of scores, as well as a selected set of verbose feedback.

5.1. Detailed Responses

Question: How do you rate the ease of use of the tool?



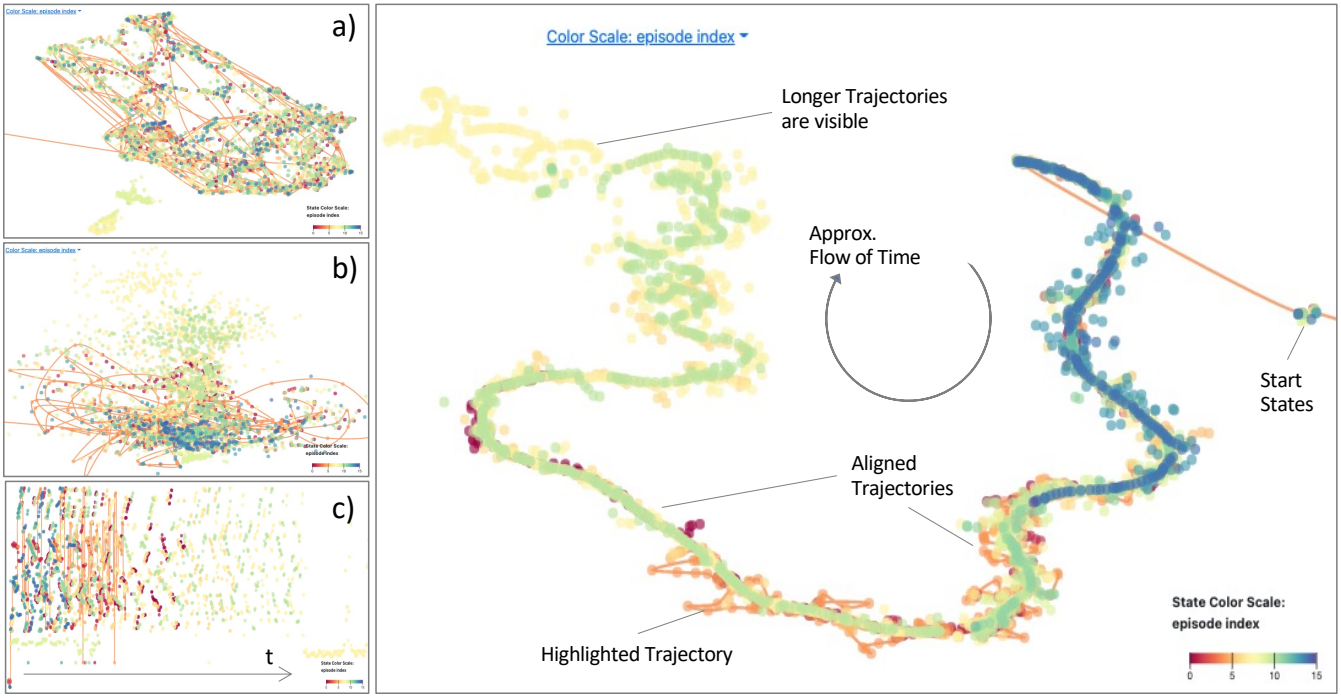
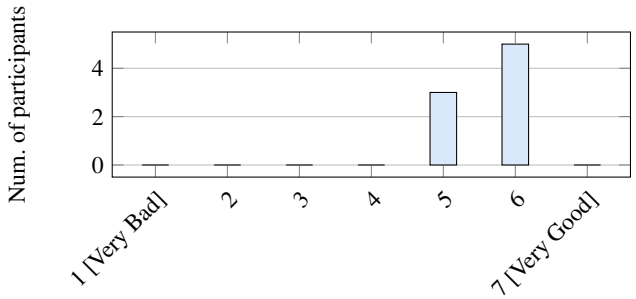
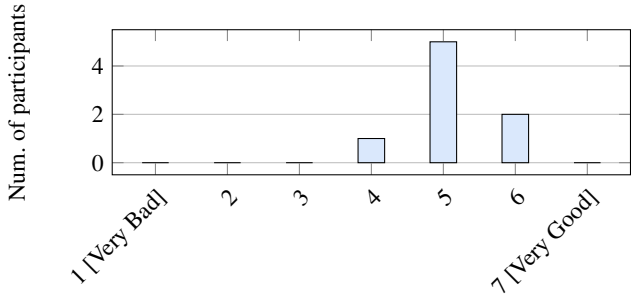


Figure 2: *The tool allows visualizing the same state sequence embedding with a different weighting of the temporal component: (a) No weight on the temporal component (b) The step-index inside an episode is concatenated to the data vector, but its magnitude is scaled to the maximum value in the data, (c) 1D UMAP embedding with the step on the x-axis, and state similarity on the y-axis, (d) the data vector is not scaled which leads to the step number being the dominant component determining step similarity.*

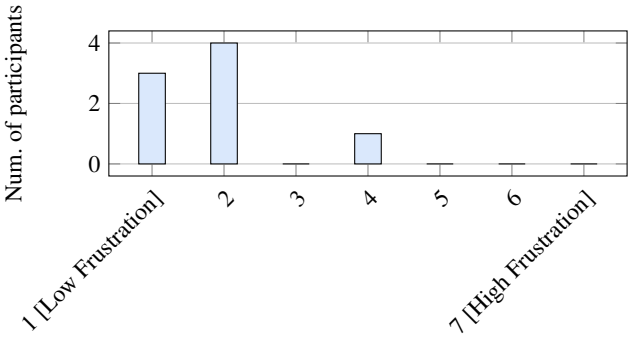
Question: How do you rate the completeness of the tool?



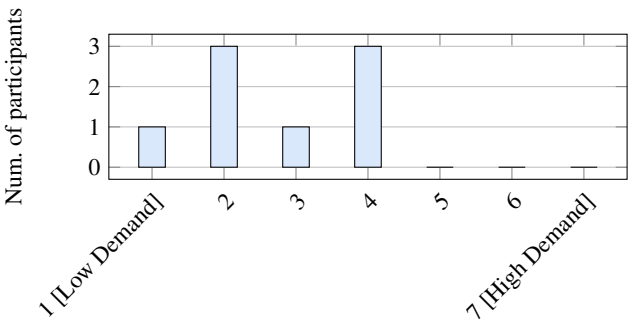
Question: How do you rate the Effectiveness of the tool?



Question: How do you rate the Frustration you feel while using the tool?



Question: How do you rate your Mental Demand while using the tool?



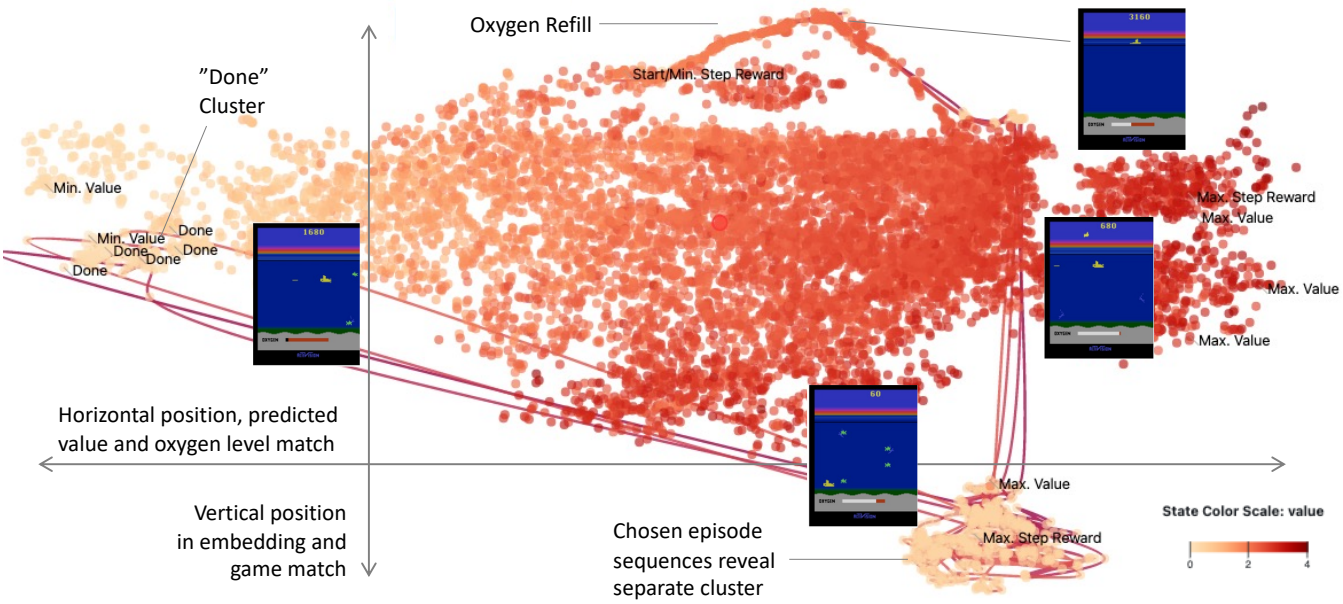


Figure 3: *The tool allows visualizing the same state sequence embedding with a different weighting of the temporal component: (a) No weight on the temporal component (b) The step-index inside an episode is concatenated to the data vector, but its magnitude is scaled to the maximum value in the data, (c) 1D UMAP embedding with the step on the x-axis, and state similarity on the y-axis, (d) the data vector is not scaled which leads to the step number being the dominant component determining step similarity.*

5.2. Selected Responses of participants

Question: Would you consider this tool useful for analyzing/training RL agents? For which tasks would you consider it to be useful? Answers:

1. Yes, specifically for tasks that do not have an easy visualization
2. Yes, it would be useful for any kind of task that includes complicated environments and actions
3. Yes, for visualization, potentially debugging

Question: What did you like most about the tool?

1. Seeing which events lead to reward spikes. Also seeing the trend in reward over time
2. Different views, annotate tool/clustering
3. Reward View, Action distribution diagram

Question: What is still missing in your opinion? Here, we focus on suggestions that were given after the second expert study, as most feedback after the first expert study was already integrated into the second prototype:

1. It's not completely clear how the projection onto 2D space is being done; including the metrics that are used to do this projection would be useful (or allowing the user to change these metrics). Also, allowing the user to change the speed of playback or use the arrow keys would be useful. *(Authors Note: The feature to change projection settings is already available in the application)*
2. Better description/guidelines for some of the embedding views
3. (The) current state seems to be complete for me

6. Additional Screenshots and Results

Here, we provide a small number of additional screenshots that showcase some of the functionalities:

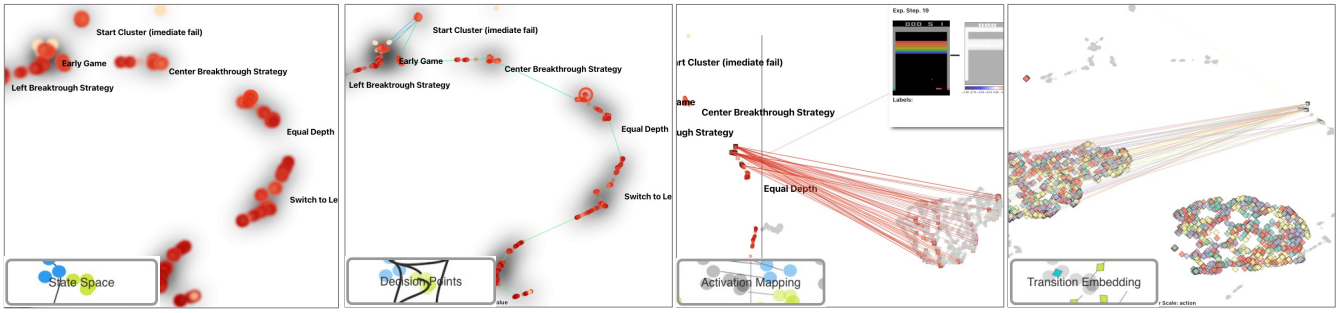


Figure 4: *The four available layer options for the state sequence embedding side by side: The **State space** shows abstracted information, in this case, the value estimate of the RL algorithm. The state has been annotated which is visible in the embedding visualization, the **decision point** view only shows a subset of relevant trajectories and states, and the **activation mapping** view shows the relationship between highlighted observations and activations, and finally, the **transition embedding** is an alternative visualization, which visually connects state-to-state transitions to respective states.*

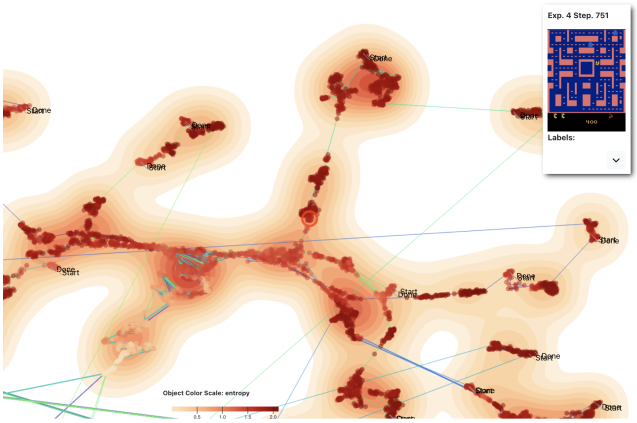


Figure 5: *State Space with a different color scale, here state entropy.*

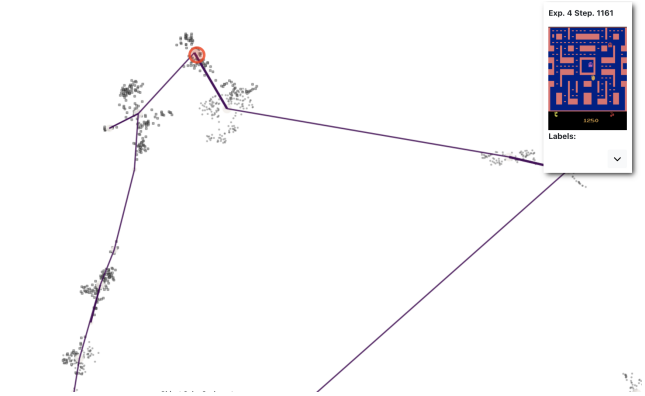


Figure 6: *Decision Point view for a non-branching trajectory. Small segments, e.g., loops, are abstracted.*

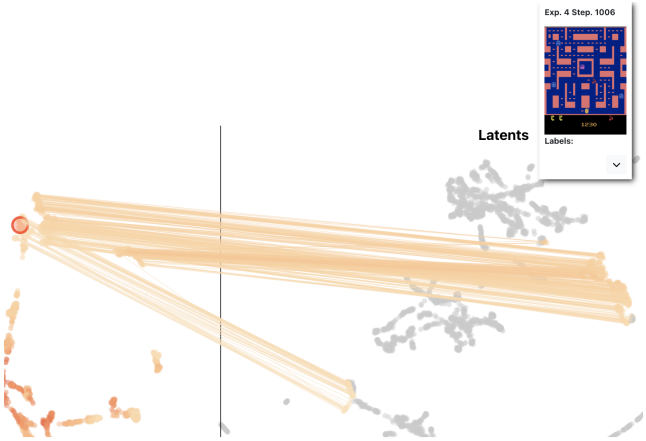


Figure 7: *Mapping between observation states and latent observations. Parallel connection lines indicate similar mapping to the state space. Diverging directions indicate similar states mapped, e.g., to different parts of the state space.*

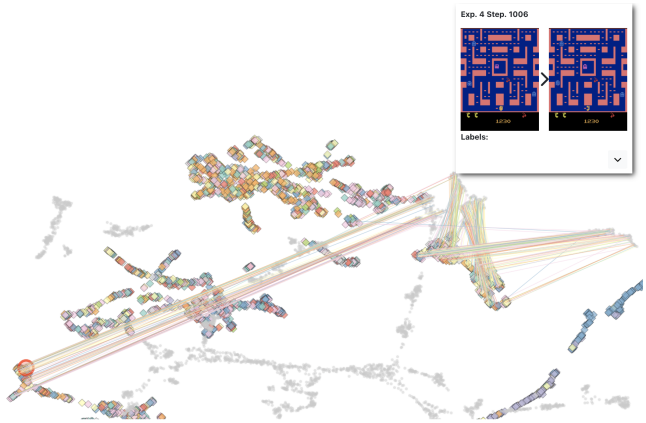


Figure 8: *Transition embedding with a larger set of highlighted states. Again, lines that deviate from the parallel pattern are visible and can indicate states that can be further investigated.*

References

[AGM*18] ADEBAYO J., GILMER J., MUELLY M., GOODFELLOW I., HARDT M., KIM B.: Sanity checks for saliency maps. In *Proc. of Intl. Conf. on Neural Information Processing Systems* (Red Hook, NY, USA, 2018), NIPS'18, Curran Associates Inc., p. 9525–9536. 1

[GKDF18] GREYDANUS S., KOUL A., DODGE J., FERN A.: Visualizing and understanding Atari agents. Dy J., Krause A., (Eds.), vol. 80 of *Proc. of Machine Learning Research*, PMLR, pp. 1792–1801. URL: http://proceedings.mlr.press/v80/greydanus18a.html. 1

[Lek] LEKSCHAS F.: Flekschas/regl-scatterplot: Scalable webgl-based scatter plot library build with regl. URL: https://github.com/flekschas/regl-scatterplot. 3

[MHM18] MCINNES L., HEALY J., MELVILLE J.: Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 2

[RHE*19] RAFFIN A., HILL A., ERNESTUS M., GLEAVE A., KANERVISTO A., DORMANN N.: Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019. 3

[RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: "why should I trust you?": Explaining the predictions of any classifier. In *Proc. of 22nd ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016* (2016), pp. 1135–1144. 1

[SG20] SEQUEIRA P., GERVASIO M.: Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence 288* (2020), 103367. 1

[SHS*20] SHI W., HUANG G., SONG S., WANG Z., LIN T., WU C.: Self-Supervised Discovering of Interpretable Features for Reinforcement Learning. *arXiv e-prints* (Mar. 2020), arXiv:2003.07069. arXiv:2003.07069. 1

[SMG20] SAINBURG T., MCINNES L., GENTNER T. Q.: Parametric umap embeddings for representation and semi-supervised learning. *Neural Computation 33* (9 2020), 2881–2907. URL: https://arxiv.org/abs/2009.12981v4, doi:10.1162/neco_a_01434. 3

[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. 1

[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

[WTGE20] WANG S., TOYER S., GLEAVE A., EMMONS S.: The *imitation* library for imitation learning and inverse reinforcement learning. https://github.com/HumanCompatibleAI/imitation, 2020. 3

[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. 1