RLMViz: Interpréter la Mémoire du Deep Reinforcement Learning

RLMViz: Interpreting Deep Reinforcement Learning Memory

Theo Jaunet, Romain Vuillemot, Christian Wolf

English Abstract—We present RLMViz, a visual analytics interface to interpret the internal memory of an agent (e. g., a robot) trained using deep reinforcement learning. This memory is composed of large temporal vectors updated before each action of the agent moving in an environment. This memory is not trivial to understand, and is referred to as a black box, which only inputs (images) and outputs (actions) are understood, but not its inner workings. Using RLMViz, experts can form hypothesis on this memory and derive rules based on the agent's decisions to interpret them, and gain an understanding towards why errors have been made and improve future training process. We report on the main features of RLMViz which are memory navigation and contextualization techniques using time-lines juxtapositions. We also present our early finding using the VizDoom simulator, a standard benchmark for DRL navigation scenarios.

1 Context

Learning to navigate in an environment (e. g., in a house, on the road) is the core challenge for moving objects (e. g., mobile robots, autonomous cars) we will refer to as *agents*. Deep Reinforcement Learning (DRL) is an efficient machine learning technique to tackle the navigation problem, by focusing on interactions between the agent and its environment [8]. Agents trained with DRL recently achieved human-level performances on many Atari 2600 games [5].

The underlying learning process in DRL is efficient as it explores large space of strategies, that even humans may not have tested. Thus one cannot just train an agent using annotated datasets by humans, combined with supervised machine learning methods: the agent needs to perform trial and errors along the way. The reward the agent receives after an action is its only way to determine the quality of its action and learn from it. However a reward is not necessary due to the last action (e. g., turn left, right) the agent did, but probably due to earlier actions. Therefore, determining how to assign credit for each action that lead to a reward remains an open research question [2].

As expressed in [1], data visualization has been proven helpful to deep learning experts to better understand their models by providing insights on their decisions and inner representations. However, to the best of our knowledge, deep learning memory visualizations have mostly been applied to text processing with work such as LSTMVis [7], and in [3]. The work the most related to ours is, DQNViz [9], which studies memory-less DQN agents behavior as they learn how to play at atari 2600 breakout. Such work demonstrates effectiveness of visual analytics solutions applied to DRL. However, DQNViz is not adaptable to agent with memory, and only works with agents moving in 1D (e. g., left or right) and our focus is on moving agents in a 2D spaces.

- *Theo Jaunet: LIRIS, INSA-Lyon*
  E-mail: theo.jaunet@insa-lyon.fr.
- *Romain Vuillemot: LIRIS, École Centrale de Lyon*
  E-mail: romain.vuillemot@ec-lyon.fr.
- *Christian Wolf: LIRIS, CITI, INSA-Lyon, Inria*
  E-mail: christian.wolf@insa-lyon.fr.

2 Navigation Problem and Memory

Our focus is on navigation problems, where an agent goal is to reach a particular location (e. g., to pick items, find a particular spot). To do so, the agent must explore its environment using discrete actions $a \in A$, with $A = \{left, right, forward, backward\}$.

The navigation problem we focus on uses ViZDoom [4] which provides instances of the navigation problem with the video game Doom (very popular in the 90's). Vizdoom provides 3D environments, and scenarios focusing on various goals (e. g., survive, reach a location, gather items). We focus on the health gathering supreme scenario, where the agent must pick health-packs (HPs) randomly placed in the environment. The urge to pick HPs is motivated by the attribute h representing the agent's health, with $h \in H[0, 100]$, which is periodically reduced. This task ends when the agent survived long enough (i. e. 525 time-steps), or when h reaches 0 (death). To complete this task, the agent uses as input the projection of its field of view as a matrix of pixels. The output is a vector of 8 values within the continuous $[0, 1]$ range and its sum equals to 1. Each value corresponds to how the agent estimates the potential of each action. Thus the agent chooses the action with the highest values, i. e. the one which may provide the highest reward.

Fig.1 shows how the memory is constructed while the agent navigates in an environment, based on its perception of the environment and its actions. The memory is a time-varying summary of what the agent previously saw, and what it currently sees.

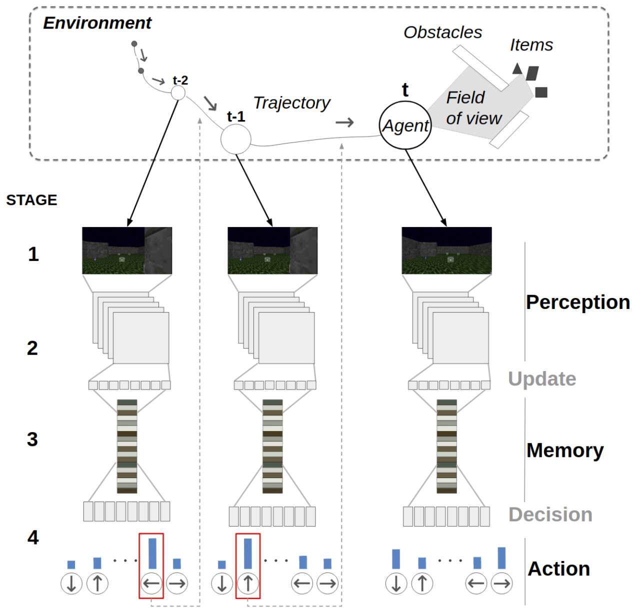

Fig. 1. As the agent discovers its environment, it produces a trajectory (top). After each action, the agent observes its environment (field of view) (stage 1) from which it perceives features (stage 2). Those features are compressed and used to update the agent's memory (stage 3). From this memory, the agent decides what action it should do (stage 4).

3 Interpreting Memory using RLMViz

Fig. 2 shows an overview of RLMViz to assist analysts in interpreting DRL memories introduced previously. Fig. 2 ① shows the fully trained memory where color encodes the activated areas, supposedly the ones involved for each decision (e. g., turn left or right). It encodes vectors (vertical column) of 512 cell over a 525 time step interval. Each cell (square) encodes a quantitative values using a bi-variate color scale. By default it shows the vector as it is produced by the agent. However it can be re-ordered and filter with respect to rules on abstractions (e. g., high confidence in actions). As this view is abstract, additional data (e. g., directions) as juxtaposed below this timelines encoding metrics (reward, heath, ..) and actions (up, right, ..). It puts them in perspective with activations above so users can see a parallel between the patterns. Contextual views are included and coordinated such as the input image, decisions and trajectory.

The main interaction in RLMViz is a vertical time slider showing the current time t_c which is shared by all the other views which are either aligned with the memory as they also represent an interval. Using This slider, analysts can select subset of the memory, replay scenario during the corresponding time interval, and find correlations between activations and metrics, and produce knowledge, by:

- *browsing* the memory (as a timeline) and check what the agent sees and its decisions; visual cues for selection are dark, consecutive patterns.
- *annotating* the timeline by selecting time interval where something may be interesting based on the activation, but also with additional timelines (actions, etc.) and set a name.
- *summarizing* how rules are related to each other times during the whole time interval (episode) using a state machine.

We are currently conducting field studies with experts in DRL to evaluate RLMViz interpret-ability power. We received promising early feedback from experts who requested to use RLMViz more frequently in their work routine (right now they only generate videos of trained agents to understand their decision). Using RLMViz, they managed to grasp insights on the agent's behavior in various situations such as facing health-packs, facing empty corridors, etc. The experts are particularly interested in how RLMViz can be applied to more realistic simulators such as habitat [6], and tasks such as "go to the kitchen".

As future work we are particularly interested in understanding how such exploratory process can be summarized using a state chart view (Fig. state charts ④) which aims at capturing the knowledge from the agent's decision, informed by its memory activation. The current version of this feature represents how timeline annotations are related but we assume it may capture higher level knowledge such as agents strategies, transferable across many scenarios.

References

[1] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. Technical report.
[2] N. Justesen, P. Bontrager, J. Togelius, and S. Risi. Deep Learning for Video Game Playing. Technical report.
[3] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing And Understanding Recurrent Networks. page 11, 2016.
[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. *arXiv:1605.02097 [cs]*, 5 2016.
[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, 12 2013.
[6] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. 4 2019.
[7] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks. *arXiv:1606.07461 [cs]*, 6 2016.
[8] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction Second edition, in progress. Technical report.
[9] J. Wang, L. Gou, H.-W. Shen, and H. Yang. DQNViz: A Visual Analytics Approach to Understand Deep Q-Networks. Technical report.

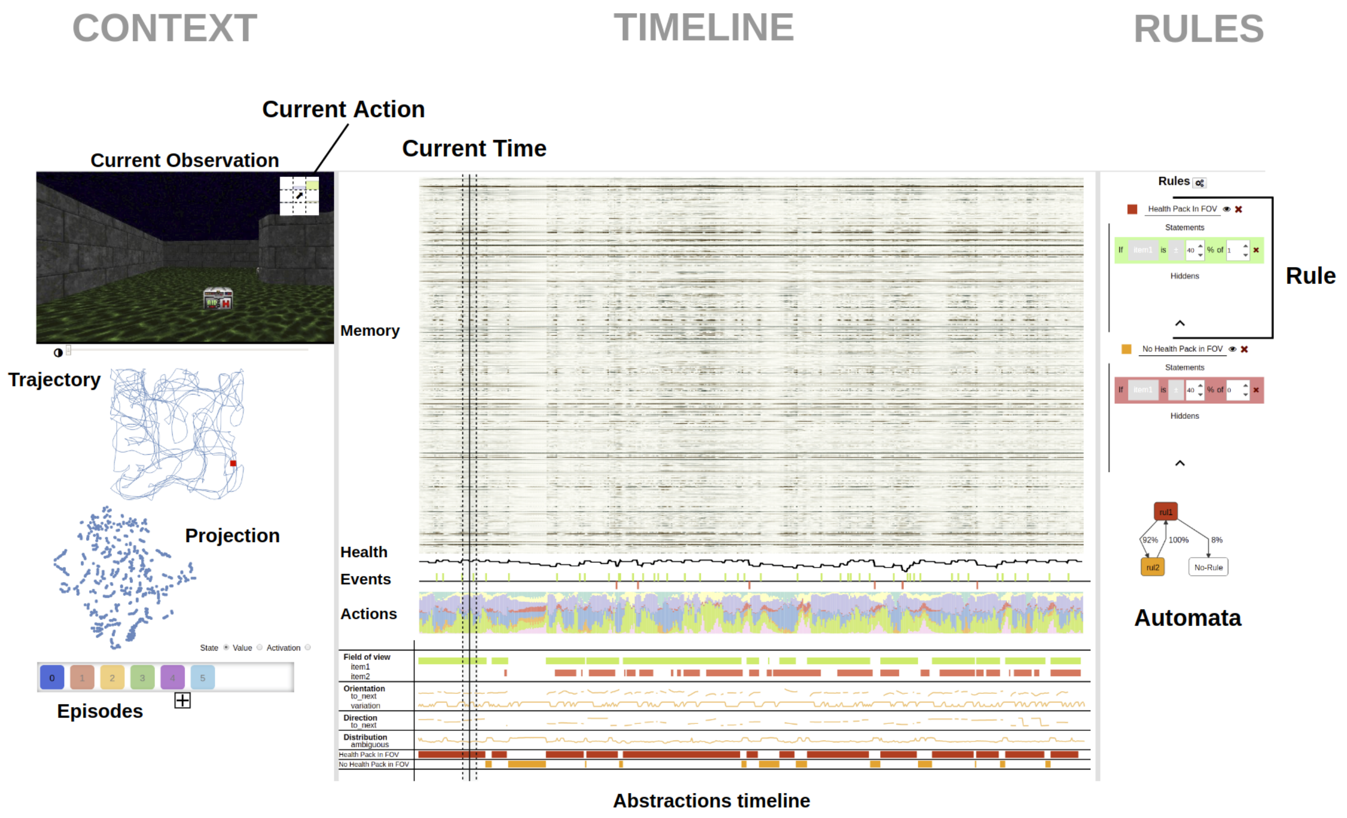

Fig. 2. RLMViz is designed to understand the memory timeline ① of an agent trained using Deep Reinforcement Learning (DRL). This memory is a large temporal vector considered as a black box and not trivial to understand. Using RLMViz analyst *browse* this timeline to find activated areas based on the agent's perception and action ②. It can then select intervals ③ to set rules and summarize them as state charts ④.