DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning

Théo Jaunet, Romain Vuillemot, and Christian Wolf

Abstract—We present DRLViz, 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 when the agent moves in an environment and is not trivial to understand. It is often referred to as a black box as only inputs (images) and outputs (actions) are intelligible for humans. Using DRLViz, experts are assisted to interpret using memory reduction interactions, to investigate parts of the memory role when errors have been made, and ultimately to improve the agent training process. We report on several examples of use of DRLViz, in the context of video games simulators (ViZDoom) for a navigation scenario with item gathering tasks. We also report on experts evaluation using DRLViz, and applicability of DRLViz to other scenarios and navigation problems beyond simulation games, as well as its contribution to black box models interpret-ability and explain-ability in the field of visual analytics.

Index Terms—Deep Reinforcement Learning (DRL), visual analytics, model interpretation, Recurrent Neural Networks (RNN).

1 Introduction

Automatic navigation is among the most challenging problems in Computer Science with a wide range of applications, from finding shortest paths between pairs of points, to efficiently exploring and covering unknown environments, up to complex semantic visual problems (*"Where are my keys?"*). Addressing such problems is important for modern applications such as autonomous vehicles to improve urban mobility, but also social robots and assisting elderly people. Historically, navigation was often solved with discrete optimization algorithms such as Dijkstra [1], A-Star [2], Front-propagation [3] etc., applied in settings where spatial maps are constructed simultaneously with solving the navigation problem. These algorithms are well understood, but are restricted to simple waypoint navigation. Recently, techniques from Machine/Deep Learning have shown spectacular progress on more complex tasks involving visual recognition, and in particular in settings where the agent is required to discover the problem statement itself from data. In particular, Reinforcement Learning (RL) and the underlying Markov Decision Processes (MDP) provide a mathematically founded framework for a class problems focusing on interactions between an agent and its environment [4]. In combination with deep networks as function approximators, this kind of models was very successful in problems like game playing [5], [6] and navigation in simulated environments [7], [8], [9], and work in human-computer interaction (HCI) emerging [10]. However, those models lack in interpretability and have a reputation of being black boxes [11] where only inputs and outputs are intelligible for humans.

Interpretability [12] can help developers [13] to build more efficient models faster, with less biases, and thus eases deployment in real life situations. Furthermore, recent laws such as the General Data Protection Regulation (GDPR) by the European Union, require to disclose the reasoning capabilities of a model or an algorithm. Interpretable models are also crucial for end-user to build trust on them and understand the process that lead to a decision.

We built DRLViz, a novel Visual Analytics interface aimed at making Deep Reinforcement Learning (DRL) models more transparent and interpretable for experts. DRLViz exposes a trained agent's memory using a set of interactive visualizations the user can overview and filter, to identify sub-sets of the memory involved in the agent's decision. DRLViz targets expert users in DRL, who are used to work with such models (referred to as *developers* in [13], [14]). Typically, those experts have already trained agents and want to investigate their decision-making process. We present examples of use for DRLViz: probing items of interest to an agent (e. g., health packs in a game, car keys or kitchen utensils for a companion robot), latent representations, observing how the agent decides between two items and interpreting its mistakes.

We also report on the use of DRLViz by 3 experts in DRL. They found the tool easy to use and useful to better improve the models by debugging current ones and understanding trained agents strategies. We conclude by detailing the main perspective of our research: reducing the memory of an agent to only its key elements. Such reduction process could help developers build not only more interpretable models as they may contain less memory dimensions, but also requiring less computing power to embed them in mobile devices (e. g., car, robots) and reduce the energy consumption footprint.

2 Context and Background

The context of our work is related to building deep neural network models to train robots achieving human assistance

- *T. Jaunet is with INSA-Lyon and LIRIS laboratory. E-mail: theo.jaunet@insa-lyon.fr*
- *R. Vuillemot is with École Centrale de Lyon and LIRIS, France. E-mail: romain.vuillemot@ec-lyon.fr*
- *C. Wolf is with INSA-Lyon, LIRIS, CITI and INRIA, France. E-mail: christian.wolf@insa-lyon.fr*

tasks in real-world environments. As the sample efficiency of current RL algorithms is limited, training requires a massive amount of interactions of the agent with the environment — typically in the order of a billion. Simulators can provide this amount of interactions in a reasonable time frame, and allow to work with a constantly controlled world, that will generate less noise (e. g., a shade) in the agent's latent representation. We will discuss in the perspectives section the extension of our work beyond simulators and the knowledge transfer from simulation to real-world scenarios, where variability (e. g., lighting, clouds, shades, etc ..) and non-deterministic behaviors (e. g., robots may turn more or less depending on its battery charge) occur.

2.1 Navigation Problem Definitions

Our focus is on navigation problems, where an *agent* (e. g., robot, human) moves within a 2D space we call *environment* (Fig. 1). An environment contains obstacles (e. g., walls), items the agent may want to gather or avoid, and is usually bounded (e. g., a room). The goal of the agent can vary according to the problem variation, but typically is to reach a particular location (e. g., gather items, find a particular spot), importantly, the goal itself needs to be discovered by the agent through feedback in the form of a scalar reward signal the environment provides: for instance, hitting a wall may provide negative reward, finding a certain item may result in positive reward. To discover and achieve the goal, the agent must explore its environment using actions. In our case, those actions are discrete and elements of the following alphabet: $a \in A$, with $A =$\{*forward, forward+right, right, backward+right, backward, backward+left, left, forward+left*\}. The navigation task ends when the agent reaches its goal, or when it fails (e. g., dies, timeout).

As the agent explores its environment, it produces a trajectory. A trajectory is a series of positions p (x,y) in a space S bounded by the environment. Those positions are ordered by time-step $t \in T$, where $t_0 < t_1 < t_n$, and the interval between t_n and t_{n+1} is the time the agent takes to act. In addition to positions, trajectories contain complementary attributes b, which may vary depending on the agent goal (e. g., number of gathered items, velocity, etc.). We call step$_t$ the combination of both the agent position p and its attributes b, at a given time-step t. Thus $step_t$ can be represented as follows $<p_t, b_t>$. The transition between steps occurs as the agent makes a decision. An *episode* groups all iterations from the first step at t_0, until the agent wins or looses at t_n.

2.2 Navigation using the ViZDoom Simulation

The simulation environment we focus to train agents navigate is ViZDoom [15] which provides instances of the navigation problem based on Doom, a very popular video game in the 90's. ViZDoom provides a 3D world and as such is a proxy problem to mobile service robotics. It provides different scenarios focusing on various goals (e. g., survive, reach a location, gather items, avoid enemies etc.). We focus on the *health gathering supreme scenario*, where the agent needs to gather health packs (HP) randomly scattered in the environment. Incentives for gathering health packs are provided by an internal attribute $h \in H[0, 100]$ representing the agent's health, which the environment decreases by 8 every 8 steps. Each HP increases h by 20, with a ceiling of 100. The goal of the agent is to maintain h above 0, until it reaches a timeout of 525 time-steps. This task ends when the agent survived until the timeout is triggered (win), or when h reaches 0 (fail). The environment contains obstacles, such as walls which are static and poison vials scattered across the environment like HPs, which, when gathered, reduce the agent health by 30: the agent should avoid them. Despite ViZDoom being a 3D world, the agent positions p are within a bounded continuous 2D plane corresponding to the bird's eye view of the environment. We summarize a step as follows: $<p_t, (h_t, s_t)>$.

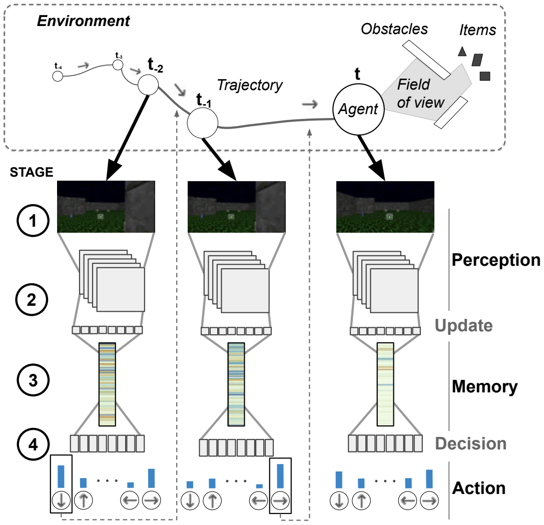

Fig. 1. Our navigation problem consists in solving a visual task (e. g., fetch, interact, or recognize items) while avoiding obstacles in an environment. Deep Reinforcement Learning can be used to solve this problem by using an image as input ① at time t. Features are then extracted from this image ②, and combined with the previous memory vector $t-1$ ③. Using this memory vector, the agent decides to move forward or turn left, for instance ④.

The task is challenging, as the agent is required to take a decision on the next based on partial information on the environment only, i.e. the task is partial observable. The observed image represents the agent's field of view (i. e. what is in front of it), in a 90 degree range and unlimited depth. The agents is required to recall previously seen observations in some way as it doesn't have access to a global view of the map, and these views are stored in its latent memory, the representation studied in this work.

2.3 Deep Reinforcement Learning and Memory

As expressed in the taxonomy [16], DRL reached state of the art performance in tasks such as robot control [17] and board games [6], [18] where it even surpasses humans in some cases. Recent Deep Reinforcement learning (DRL) models, such as Deep Q-networks (DQN) [5], [19], and Asynchronous Advantage Actor-Critic (A3C) [20], learned to play video games with human level control using only images as input. As a result, they achieved human-level performances on Atari 2600 games [21] such as breakout. Those models rely on the hypothesis that the optimal action can be decided based on a single frame.

However, these approaches operate on environments that can be totally observed (like a chess or GO board game), and not partially with a field of view which is smaller than the environment. To address this, an internal latent memory can be introduced [22] to provide a space the model can use to store an approximation of the history of previous inputs and solve navigation problems [23], [24], [25], allowing learning in simulated environments such as Matterport3D [26], ViZDoom [15], [27].

2.4 Visual Analytics and Deep Learning

Visual Analytics have been proven significantly helpful to deep learning (DL) experts to better understand their models [13], by providing insights on their decisions and inner representations. Visual analytics contributions, such as LSTMVis [14] allows users to formulate hypothesis on the memory behavior with respect to the current input sentence by selecting time intervals, and observing the memory activity through proximity search across different views and metrics. The re-ordering of memory elements using a 1D t-SNE projection applied to handwriting trajectory prediction [28] provides an overview of the representation and to highlight patterns on how different feature dimensions reacts to different path e. g., curvatures. Memory dimensions displayed over the input text of a character level prediction model [29] highlights characters that trigger specific memory activations, and thus provide insights on how certain parts of the memory react to characters (e. g., to quotes).

Despite being effective in providing insights on their models decisions and inner representations, these works cannot be directly applied to deep reinforcement learning (DRL), in particular in complex 3D environments. In these problems, the input are images, which are embedded in high dimensional spaces. Other factors are the lack of supervision over actions, known as the credit assignment problem inherent to RL problems (the reward provided at a given time step can correspond to decisions taken at arbitrary time steps in the past) and the partial observability of the problem. Some of the work has been applied to natural language processing in a supervised context, where ground truth is provided, and cannot be directly applied to DRL.

To our knowledge, DRL visualizations are under represented in the literature compared to other methods on visualizing deep learning. LSTM activations from an A3C agent [23] have been displayed using a t-SNE [30] projection. Despite being effective for an overview of the agent's memory, it offers limited information on the role of the memory. t-SNE projections have also been applied to memory-less DRL agents on 2D Atari 2600 games, in the seminal DQN paper [5], and in [31]. Recent efforts to visualize DQN such as DQNViz [32] study memory-less agents behaviors as they learn how to play the fully observable Atari 2600 breakout game. It demonstrates the effectiveness of visual analytics solutions applied to DRL but again is not directly adaptable to agents with memory, and only works with agents moving left or right in a 2D environment.

In this paper, we address the under-explored challenge of visualizing a model's memory in a self-supervised and RL context, and the visualizations of DRL agents behaviors in navigation problems. We also provide interaction to overview, filter, and select parts of such memory to provide clues on agents decision reasoning.

3 Model and Design Goals

This section presents the model we used to design and implement DRLViz. We describe the inner workings of those models and data characteristics. One key aspect being how the memory of DRL is created and updated by the agent, over space and time. Note that those data will be generated and then visualized with DRLViz after the training phase.

3.1 DRL Model

The DRL model we relied on only receives pixels from an RGB image as input, from which it decides the action the agent should perform with the Advantage Actor-Critic (A2C) [20] algotrithm. The model is composed of 3 convolutional layers followed by layer of Gated Recurrent Unit (GRU) [33], and Fully Connected (FC) layers to match the actions set A. This model is inspired by *LSTM A3C* as presented in [23] with A3C instead of A2C, and a LSTM [34] instead of GRU. Those changes reduce the agent's training time, while preserving its performances. The underlying structure that allows our model to associate raw pixels to an action is illustrated on Fig. 1 and described as follows:

Stage 1: Environment → Image. First, the agent's field of view is captured as image x_t, i.e. a matrix with dimensions of 112×64 with 3 RGB color channels.

Stage 2: Image → Feature vector. The image x_t is then analyzed by 3 convolutional layers designed to extract features f_t, resulting in a tensor of 32 features shaped as a 10×4 matrice. These features are then flattened and further processed with a Fully Connected (FC) layer. Formally, the full stack of convolutional and FC layers is denoted as function $f_t = \Phi(x_t, \theta_\Phi)$ with trainable parameters θ_Φ taking x_t as input and given features f_t as output.

Stage 3: (Features + previous memory) → New memory. The model maintains and updates a latent representation of its inputs using a Gated Recurrent Unit (GRU) [33], a variant of recurrent neural networks. This representation, called hidden state h_t, a vector of 512 dimensions, is time varying and updated at each time-step t with a trainable function Ψ taking as input the current observation, encoded in features f_t, and the previous hidden state h_{t-1}, as follows: $h_t = \Psi(h_{t-1}, f_t, \theta_\Psi)$.

Stage 4: Memory vector → Action. The model maps the current hidden state h_t to a probability distribution over actions A using a fully connected layer followed by a softmax activation function, denoted as the following trainable function:

$a_t = \xi(h_t, \theta_\xi)$ with trainable parameters θ_ξ. The highest probability corresponds to the action a_t which the agent estimated as optimal for the current step t.

The full set of parameters $\theta = \{\theta_\Phi, \theta_\Psi, \theta_\xi\}$ is trained end-to-end. During training, the agent does not necessary choose the action with the highest probability, as it needs to explore its environment, and eventually find better solutions. However, once the agent training is done, it always choose the action with the highest probability.

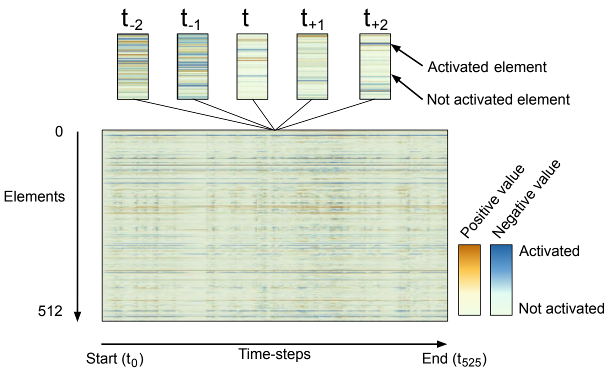

Fig. 2. Memory construction process: at the current time-step t, the agent updates its memory by producing a new memory vector. Each dimension of this vector (represented as a column) is appended to the previous ones chronologically (from left to right). As a result, each row of the appended vectors represent the actions of a single memory element.

3.2 Constructing the Memory of DRL

In the partially observed navigation problem we focus on, the agent only sees the current observation, i. e. what is in its field of view at the time-step t. However, past observations are also relevant for decision making (e. g., to gather previously seen items). Therefore the agent needs to build a representation of relevant information extracted from the history of observations. This information is encoded in h_t, a high dimensional (512 in our case) time varying vector.

Fig. 2 represents the construction process of the hidden states matrix, which consists of the hidden states h_t over the time of an episode — the central visualization in DRLViz (Fig. 3). Each hidden state is vertically aligned per time-step t at which they are produced. Therefore, the accumulation of hidden states forms a large 2D matrix, where the horizontal axis is time ($ht-1 < ht < ht+1$) and the rows are elements. A row of this 2D matrix represents the evolution and activity of a hidden state element through time and space as the agent moves. The activity of a hidden state element is characterized by its value. In our case, each element of the hidden states is a quantity within the range $[-1, 1]$. A value close to 0 represents low activity, whereas a value close to any extremity represents high activity. As it can be seen in Fig. 2, hidden states can drastically change their values between two time-steps. Such value changes can be widely observed across hidden states elements during episodes. However, it remains unclear which elements, correspond to which representations, and thus, responsible for decisions.

4 Design of DRLViz

We built DRLViz as a visual analytics interface to understand the connections between the latent memory representation (as depicted in Fig. 2) and decisions of an agent trained using Deep Reinforcement Learning. DRLViz primarily exposes the internal memory (Fig. 3) which is interactive and allows *overviewing*, *filtering* and *reduction* both for exploration and knowledge generation [35]. DRLViz is designed towards experts in DRL to identify elements responsible for both low-level decisions (e. g., move towards a spotted HP) and eventually higher-level strategies (e. g., optimizing a path).

4.1 Design Motivation and Goals

We iteratively built DRLViz with frequent meetings from colleagues experts in DL and DRL (a total of 12 meetings with 3 experts over 7 months). Our motivation to build DRLViz was to support their current manual and repetitive exploration of properties of a trained agent. They currently record videos to playback agents episodes to visualize inputs (video feed including the point of view of the agent) and outputs (actions probability) and get a sens of the strategy. Our approach was to re-build a similar interface with input/output views and facilitate playback that include interactivity as baseline. We primarily report on advanced views and interactions we added to support advanced models visualization aimed at models developers [13].

Based on a review of current practices researchers in deep learning have and related work, we identified the following design goals (**G**) to be addressed to understand the behavior of a trained agent using a learning model:

- **G1 Show an agent's decisions over (a) space and (b) time**, especially input and outputs of the model.
- **G2 Expose the memory's internal structure**, i.e. the temporal vector described in Fig.2.
- **G3 Compare memory over (a) time and (b) decisions** with multiple endpoints, e. g., starting with a specific time point, memory or trajectory behavior.
- **G4 Identify a sub-set of the memory (a sub-space)** tied to a specific agent behavior or strategy.

The high-level goal of DRLViz is to link the agent behavior to elements in the memory (rows in the memory timeline), over different episodes to make sure they are independent from the specific input. At the end of the exploration process using DRLViz, experts should be able to understand and explain most of the behaviors learned by the agent.

4.2 Overview and Workflow of DRLViz

Fig.3 shows an overview of DRLViz where the most prominent visualization is the memory timeline of a trained agent (**G2**). The primary interaction is browsing the timeline and playback the input video feed and action probabilities (**G1**). Beyond re-playing scenarios, DRLViz implements multiple interactions to:

1) *Overview* the memory and check what the agent sees and its decisions; visual cues for selection are dark, consecutive patterns (Fig. 2).
2) *Filter* the timeline when something is of interest, e. g., related to the activation, but also with additional timelines (actions, etc.).
3) *Select* elements whose activation behavior is linked to decisions. Those elements are only a subset of the whole memory and are visible on Fig. 3 ④.

Those interactions are carried out using a *vertical thumb* similar to a slider to explore time-steps t and select intervals. Such a selection is propagated to all the views on the interface, whose main ones are *image (perception)* and *probabilities (of actions)* which provide context on the agent's decisions (**G1** (b)). The input image can be animated as a video feed with the playback controls, and a saliency map overlay can be activated [36] representing the segmentation of the image by the agent. The *trajectories* view (Fig. 3)

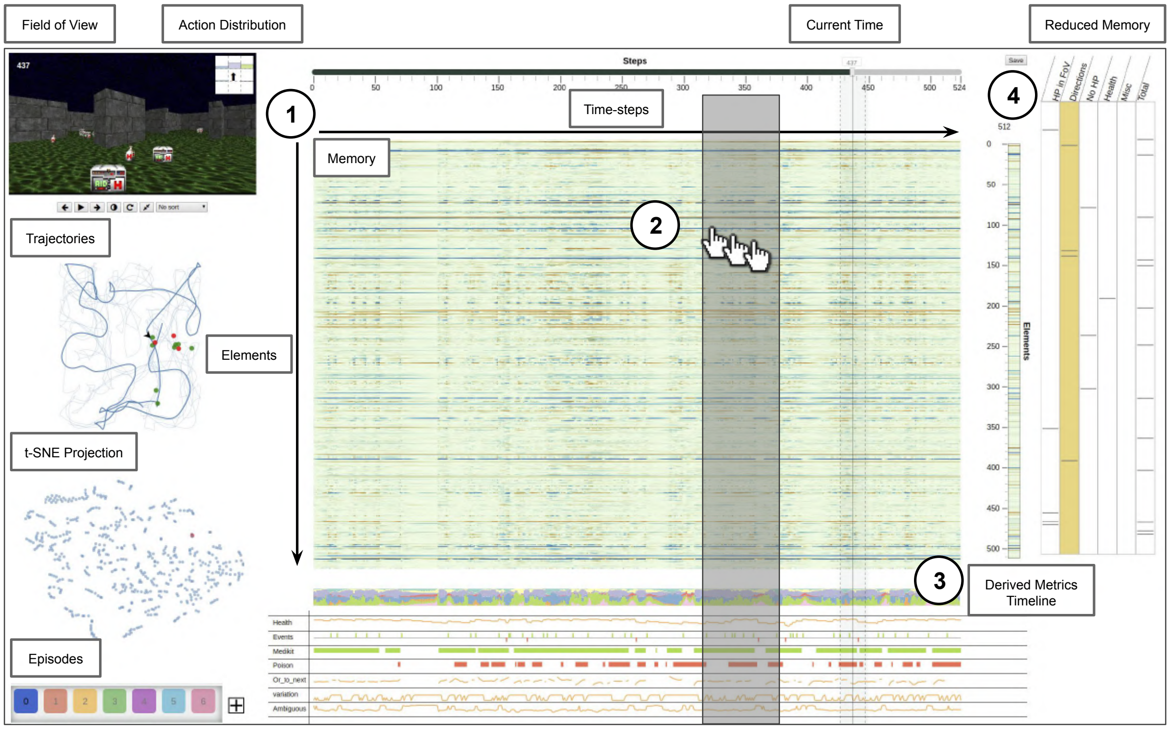

Fig. 3. DRLViz primarily display the trained agent memory, which is a large temporal vector (mapped *vertically* on ①) updated over time (mapped *horizontally* on ①). Analysts can *overview* this memory linearly in a temporal way; *filter* according to movements of the agent and derived metrics we calculated ② (e. g., when an item is in the field of view ③); and *select the memory* to filter elements to the one explaining decisions ④.

displays the sequence of agent positions $p_{t-1} > p_t > p_{t+1}$ on a 2D map (**G1** (a)). This view also displays the items in the agent's field of view as colored circles, green for health packs and red for poison vials. The position p_t, and orientation of the agent are represented as animated triangle. The user can brush the 2D map to select time-steps, which filters the memory view with corresponding time-steps for further analysis (**G3** (a)). DRLViz, also includes a *t-SNE* [30] view of time-steps t using a two-dimensional projection (Fig. 3 bottom left). t-SNE is a dimensionality reduction technique, which shows similar items nearby, and in this view, each dot represents a hidden state h occurring in a time-step t. The dot corresponding to the current time-step t is filled in red, while the others are blue. The user can select using a lasso interaction clusters of hidden states to filter the memory with the corresponding time steps.

The result of such an exploratory process is the subset of elements of the memory (rows) that are linked to an agent's decision (Fig. 3 ④). This addresses the design goal **G4**. Such subset can be seen as a memory reduction which can be used as a substitute to the whole memory (we will discuss it in the perspective sections). This subset can also be used in other episodes (with different training data) than the one used during their identification.

4.3 Memory Timeline View

The *memory timeline* exposes the memory's internal structure (**G2**), which is vector (vertical column) of 512 dimensions over 525 time-steps from which an interval can be brushed for further analysis. Each cell (square) encodes a quantitative value, whose construction is illustrated in Fig. 2, using a bi-variate color scale from [37] with blue for negative values and orange for positive values. Preserving the values as they were initially produced by the model is pertinent as some memory elements (rows) can have both positive and negative values, which may not have the same signification for the model and thus cause different decisions. This will be further explored in Sec. 6.1.

TABLE 1
List of re-ordering criteria as they appear in DRLViz. t is the current time-step, n the number of steps (525 at most), and i the element.

Criteria	Formula	Description
ACTIVATION	$\sum_{t=1}^{n} \lvert h_{ti} \rvert$	Elements most involved in decisions.
CHANGE	$\sum_{t=1}^{n-1} \lvert h_{ti} - h_{t+1i} \rvert$	Maximal change.
STABLE	CHANGE^{-1}	Minimal change.
SIMILAR	$\lvert \frac{1}{n} \sum_{t=1}^{n-1} h_{ti} - \frac{1}{k} \sum_{t=1}^{k-1} h_{tj} \rvert$	Elements in average different during an interval of k time-steps than outside it.

By default DRLViz displays the vector as it is produced by the model, hence the order of elements has no particular semantic. The memory can be re-ordered using a drop-down menu according to comparison criteria listed in table 1. In addition of those criteria, we provided the possibility to re-order the memory as it is presented in [28] i. e. a one dimensional t-SNE projection of the absolute values. The re-ordering can either be applied to the whole memory or a selected interval. An order is preserved across memory filters and episodes until the user changes it.

TABLE 2
List of *derived metrics* (implemented from top to bottom on Fig. 3 ③)

Metric	Data Type	Values
Health of the agent	Quantitative	death **[0,100]** full
Event (item gathered)	Binary	PV **(0, 1)** HP
Health pack in FoV	Binary	no HP **(0, 1)** HP
Poison vial in FoV	Binary	no PV **(0, 1)** PV
Orientation to items	Degree	left **[-45,45]** right
Variation of orientation	Quantitative	no change **[0,30]** changed
Decision ambiguity	Ratio	confident **[0,1]** indecisive

4.4 Derived Metrics View

The derived metrics timeline addresses the design goals **G3** and **G4**. It represents metrics calculated from ground truth information provided by the simulator. Those metrics aim at supporting the user find interesting agent behaviors such as *What does a trained agent do when it has no health pack in its field of view?*. The view encodes measures of both the inputs (e. g., health pack is spotted) simulator (e. g., reward, health) and outputs (e. g., up, right). Those metrics are below the memory timeline and share the vertical thumb from the memory slider (**G3** (a)) to facilitate comparisons between the memory and the behavior of the agent (**G3** (b)). The metrics can be dragged vertically by the user as an overlay of the memory to compare metrics with activation values, and identify memory elements related to them (**G4**). We provide a full list of those metrics in table 2. Two metrics are particularly complex and described as follows:

Variation describes how the the agent's orientation (i. e. its FoV) changes over time: the more it varies during an interval of 2 consecutive steps. High variations indicate hesitation in directions and intervals during which the agent turns around, whereas low variations indicate an agent aligned with where it wants to go. However, in some cases (e. g., the agent stuck into a wall), actions may have no effect on the agent's orientation which lead the variation to remain low. The maximal variation between two actions is 15 degree, thus the variation is bounded in the range $[0, 30]$.

Ambiguity of a decision is computed using the variance V of action probabilities. The variance describes how uniform actions probabilities are with respect to the mean. A variance $V = 0$ indicates that there is no difference between actions probabilities, and hence that the agent is uncertain of its decision. In the other way, a high variance represents strong differences in the actions probabilities and the agent's high confidence on its decision. Since the sum of all actions probabilities equals to 1, the variance is bounded within the range $[0, 1]$. To ease the readability, the variance is inverted as it follows: $ambiguity = 1 - V$. An ambiguity close to 1 represents an incertitude in the agent's decision.

5 Implementation

To explore the memory of a trained agent, one needs to create *instances* of exploration scenarios. For examples of use (Sec. 6) and experts evaluations (Sec. 7), we used a trained agent to explore 20 times the environment with different setups (i. e. positions of items and start position of the agent). During those episodes, we collected at each time-step information from the agent such as its FoV image, action probabilities, memory vector, and information from the environment such as the items in the agent's FoV, the position of the agent, the agent's orientation and its health. The collected data is formatted as a JSON file which groups data elements per episodes and then per steps with an average of 30Mo per episode. Those data are generated using DRL models implemented in Pytorch [38], and formatted in Python 3. More technical details are provided as supplemental material.

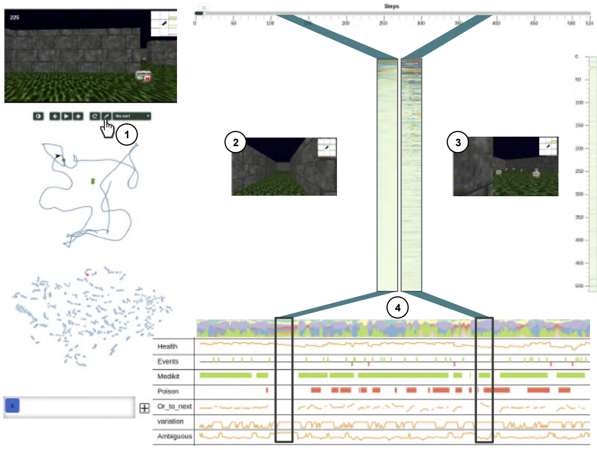

Fig. 4. DRLViz allows to compare selected time intervals ①. For instance to compare when agents face dead-ends ② and when they face health-packs ③. One can observe that more elements are active while the agent is facing HPs than while facing a dead-end. Perhaps those elements are encoding information concerning HPs. When facing a dead-end, both the orientation variation and decision ambiguity are high which can be interpreted as the agent hesitating on which action to choose.

The user interface of DRLViz loads those trained data using JavaScript and D3 [39]. The interactions between the model and the front-end are handled by a Flask Python server. The data, separated per episode is generated in a plug-in fashion i. e. without altering the model nor the simulator. Both the interface code source [1] and an interactive prototype [2] are available online.

6 Examples of Use

This section presents examples of use for DRLViz to explore agents memory on several episodes. Those examples rely on the ViZDoom *health gathering supreme* navigation problem which consists in interacting with different items in the environment, in particular in gathering HPs (Health Packs) and avoiding PVs (Poison Vials). We highlight how the use of DRLViz enables to identify memory elements involved in key decisions when navigating the environment. We picked these examples as they represent simple, yet frequent decisions made by agents and behaviors of interest for users. In the next section we report on experts feedback for more advanced use cases.

6.1 Item Representation in Memory

The primary goal of a trained agent is to gather health packs (HPs), and since they are randomly scattered across the environment, the agent must first discover them. We assume

[1] Source code: https://github.com/sical/drlviz
[2] Demo: https://sical.github.io/drlviz/

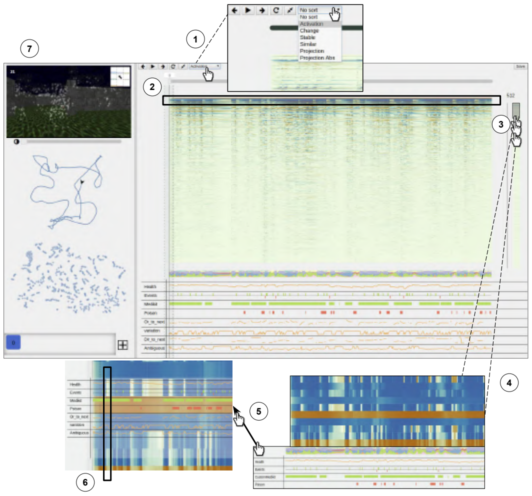

Fig. 5. The user re-orders the memory timeline to highlight the most activated elements ①. As a result, top rows ② are active while most are inactive (gray). The user can apply some brushing ③ to zoom on the top rows ④. By dragging the derived metrics timeline over the memory ⑤ the user check activated elements when health packs (HPs) are visible. One can observe an interval during which the agent has a HP in its FoV which is not represented in its memory ⑥ nor on the saliency map ⑦.

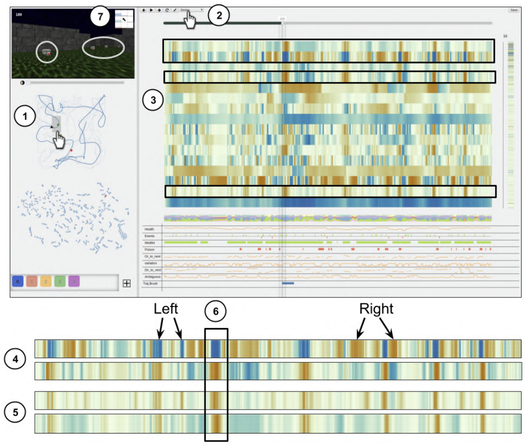

Fig. 6. Brushing the trajectory filters the memory steps based on the agent's positions ①. This memory is then re-ordered using the SIMILAR criteria ②. In the resulting memory ③, 2 elements had activations matching the agent's actions ④. While these elements reached dark values, 2 other elements ⑤ were active. During their longest activations ⑥, the agent had to decide between health packs ⑦.

that once a HP enters the agent's field of view (FoV), it influences its behavior (e. g., it moves towards it). DRLViz can be used to check this assumption and probe how discovered HPs activate (or not) hidden states dimensions.

Fig. 5 ① shows the user **re-ordering the memory timeline** per ACTIVATION during the whole episode. The ordered matrix now ranks the most activated elements at the top of (Fig. 5 ②) which usually are darker than others. A first observation is that just less than 50 elements were highly active during the whole episode. The others were either inactive (gray), or occasionally active with dark colors for short intervals of around 2 to 8 steps.

DRLViz allows to **zoom into the memory** to inspect only the top activated elements (Fig. 5 ③). Among them, some had distinguishable activations patterns e. g., from blue (activated) to gray (not activated) (Fig. 5 ④). The user can contextualize them with the **derived metric *HP in FoV* overlaying the memory** (Fig. 5 ⑤). One noticed inactive elements when there was no HP in the agent's FoV and active otherwise. This suggests that there may be a link between the metric and the memory elements, and thus that the agent may have memory elements devoted to HPs representation. However, despite having a HP in FoV, those elements were inactive during an interval of 5 steps. After **replaying** it and observing the **saliency maps** (Fig. 5 ⑥), the user noted that they were not oriented towards the HP. As the saliency map starts to highlight the HP, the elements were blue (active). Therefore one can draw the hypothesis that the agent did not perceive the HP during this interval.

6.2 Linking Memory and Items to Decisons

When multiple HPs are in FoV, the agent has to decide which one to gather first.

As HPs are tied to locations, **the geo-map with trajectory** enables to select them and identify a specific time interval in the **memory** (Fig.6 ①) with multiple HPs. While applying **playback** to such interval, the user noticed that the agent had 3 HPs in its FoV reachable in less than 8 steps (Fig.6 ⑦). However, the agent decided to turn towards the opposite direction (left) and reached HPs that were not in its FoV. One may interpret such behavior as the agent gathering previously seen HPs.

To probe hidden states elements which may be responsible for such behavior, she used the previously brushed interval to **re-order the memory with the SIMILAR criteria**. In the zoomed and re-ordered memory (Fig. 6 ③), she isolated 2 particular elements, one blue when the agent turned left, orange when the agent turned right, and its opposite i. e. orange instead of blue and blue instead of orange (Fig. 6 ④). This suggests that some elements of memory may be directly related to the agent's actions.

Despite having HPs in its FoV (Fig.6 ⑦), the agent decided to turn left and reached previously seen HPs. During this interval, the two identified elements corresponding to actions, had high activations with values assimilated to *left* actions (Fig.6 ⑥). And two other hidden states elements were active (orange) (Fig.6 ⑤). When such situation occurred, the agent either had to choose between at least two HPs in opposed directions, or aligned itself towards a HP despite having others in its FoV. The user formulated the hypothesis that the agent was able to associate actions towards HPs regardless of their presence in it's current FoV.

6.3 Parts of the Memory Linked to Erroneous Actions

Agents have been observed gathering poison vials (PVs), which should have been avoided as they may lead to death. This example covers how a user can investigate why such a mistake has been made. Contrary to the previous example, she used the **derived metrics timeline** as entry point, and started from the PV gathered to investigate the steps leading to such decision in a backward way (t, t_{-1}, ..).

The derived metrics timeline provides an overview on many episodes to spot rare events such as death by gathering PVs. After **loading all episodes** the user identified only

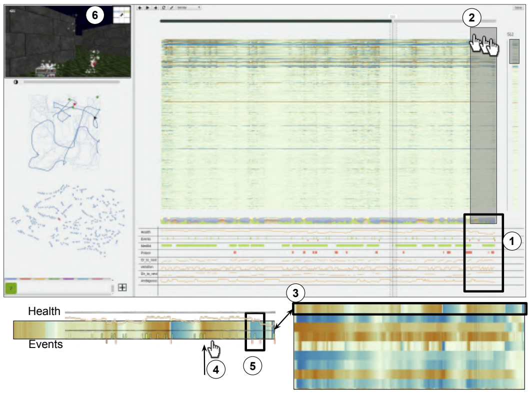

Fig. 7. DRLViz allows to detect agent deaths ① and select the corresponding steps via brushing ②. In the memory re-ordered with the SIMILAR criteria ③ and zoomed, one particular element changed to blue as the agent gathered PVs. This can be observed while overlaying the derived metrics health and events over this element ④. However, the penultimate PV gathered ⑤ did not changed the element value despite the saliency map ⑥ showing that the agent noticed the poison vial.

2 where this event occurred (Fig.7 ①). Looking at the **video playback**, the user realized the agent had enough space to dodge the PV but did not. Multiple reasons could explain it, either the agent under-estimated the space needed to avoid it, or its effect (i. e. health reduction), or just it did not noticed it at all. In **saliency maps** (Fig.7 ⑤) one observed that the pixels corresponding to PVs were highlighted. Therefore, one interpreted that the agent saw the PV but for some reasons did not avoid it.

If the agent saw the PVs but still gathered them, it either failed to avoid them, or neglected their effects. Fig. 7 ① shows the user using the derived metrics timeline to **select via brushing** (Fig. 7 ②) the time interval during which the agent gathered the 3 PVs that lead to its death. From this interval, she **re-ordered the memory with the SIMILAR criteria**. The first element among the re-ordered memory, has low activations (light orange) for most of the episode, expect few changes to blue when the agent gathered PVs. Once this element changed to blue, it gradually turned to gray and then to orange. One may interpret such element activity as encoding the agent's health. During both training and evaluation the agent did not have access to its own health, which suggests that the agent learned itself the effects of the gathered items.

Since the agent has seen the PVs, and encoded their effects in its memory, each PV gathered should be represented as drop to blue in the identified memory element. However, as observed in (Fig. 7 ④) the penultimate PV gathered by the agent did not cause a drop to blue despite the agent's low health ($h < 20$). Using **playback**, the user observed that the agent tried to avoid the PV, but failed when the PV was no longer in its FoV. Therefore, she suggested that the agent considered that he had dodged the PV, and hence did not change its health representation. With such erroneous information in its memory, the agent gathered another PV 20 steps after with low health and died. One can draw the hypothesis that the agent gathered the latest PV on purpose as it was gathering a HP at same time and had in its memory the information that it had enough health to do it.

7 EXPERTS EVALUATION

The previous examples demonstrate how simple questions can be easily addressed using DRLViz. We conducted an evaluation with experts to validate the usability of DRLViz, as well as its ability to answer advanced questions. We collaborated with a total of 3 experts in DRL which are experienced researchers building DRL models and match the target profile for DRLViz [13]. This section reports on their use of DRLViz, and insights they discovered using it.

7.1 Protocol and Custom Navigation Problem

We recruited three DRL experts (Expert #1, Expert #2, Expert #3) from two different academic laboratories to evaluate DRLViz. They were shown a demonstration of DRLViz on the *health gathering scenario* for 10 minutes. The evaluation started with DRLViz loaded with data extracted from a model and scenario both developed by Expert #1, and ended after 35 minutes. The model used had a smaller memory with 128 dimensions instead of 512 for the model used in examples. This allows the model to converge faster, and reduce the agent's training time. The navigation problem was also different than in the examples as it required the agent to provide more complex reasoning capabilities, and thus to encode more information in its memory. The agent had to achieve a *k-items* navigation problem, in which, the goal is to gather $k = 4$ items in a predefined order [27]. The agent, walls and items are randomly placed in an environment at the beginning of an episode. As opposed to the *health gathering*, the agent has no prior information on the walls as their positions changed at each episode. To succeed the agent must first gather a green armor, then a red armor, followed by a health pack, and finally a soul-sphere (blue circle). Gathering the items in another order instantly kills the agent and ends the episode (fail). The agent should use its memory to encode, both, information on the items it gathered, and the positions of items or walls encountered in order to find them quickly when the correct order has arrived and they have been previously observed. We report on feedback from the experts as well as our observations on how participants used DRLViz.

7.2 Feedback from Expert #1

Expert #1 is the most experienced expert for this evaluation as he designed both the model and the navigation task [27] and created animations of agents behaviors. Expert #1 was our primary collaborator to design and build DRLViz.

Fig. 8 shows DRLViz loaded with the *k-item* scenario. Expert #1 first selected an interval corresponding to the agent searching and gathering the last item. This interval started one step after the agent gathered the HP (third item), and ended as the agent gathered the soul-sphere (last item). Expert #1, then used the CHANGE criteria to re-order the interval. While replaying it, he noticed two elements with similar activations (Fig.8 ⑥). Those elements remained blue during the interval, however they were inactivated (gray) during the rest of the episode. With further investigation, Expert #1 noticed that those elements were active 4 steps before the agent gathered the HP. Expert #1 described those elements as *flags* i. e. elements that encodes binary information. Expert #1's intuition was that the agent learned to

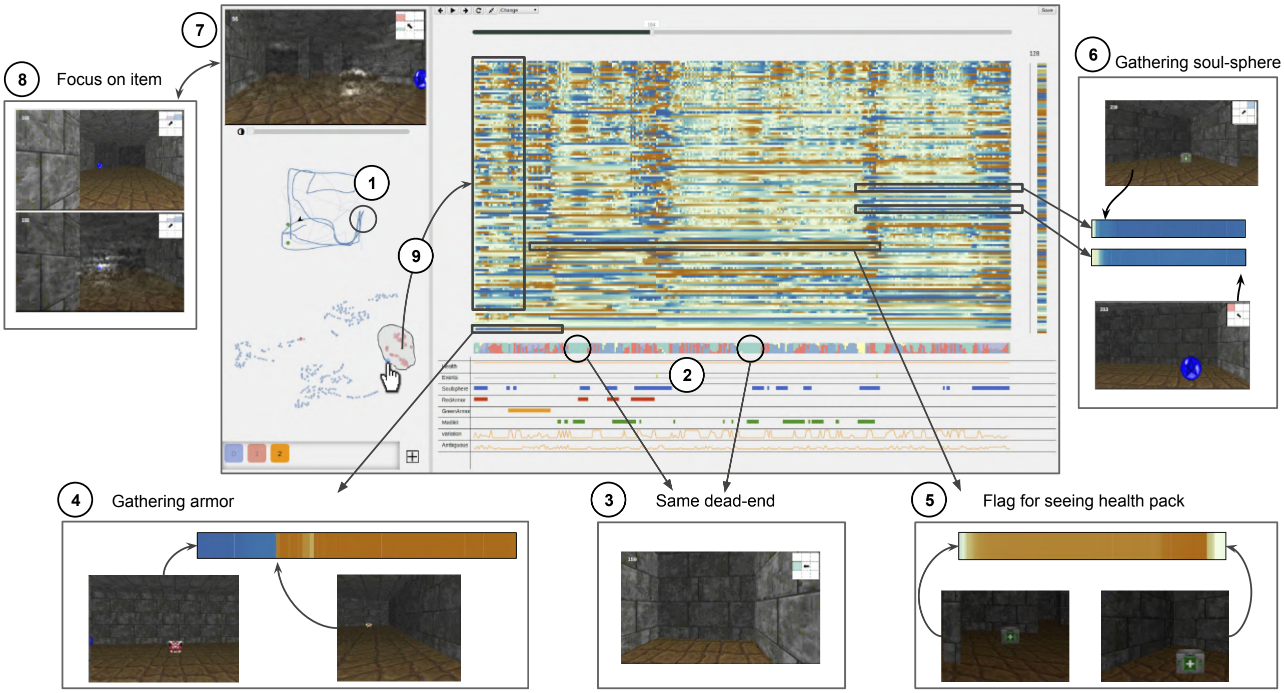

Fig. 8. Summary of the insights gained by the experts. In the trajectory ① and stream-graph of actions ② Expert #1 noticed two intervals during which the agent only turned right. After replaying those sequences, Expert #1 stated that the agent came twice in the same dead-end ③. Expert #3 observed a hidden state dimension which is blue when the agent sees the red armor before the green armor, and then remained orange until when saw the green armor ④. Expert #2 probed a dimension that is active as the agent first saw the HP, and remained active until it gathered it. Expert #1 also identified two hidden state elements that changes as the agent gathered the health pack and then kept their values until the end of the episode ⑥. Using saliency maps ⑦, Expert #2 observed that the agent ignore the soul-sphere until it gathered the 3 firsts items ⑧. Finally, Expert #3 identified clusters in the t-SNE projection which corresponds to the agent's objectives e. g., gathering the green armor ⑨.

complete the navigation problem by focusing on one item at the time. And that it only used its memory to encode information on items it already gathered, and hence which item it should currently gather. **Expert #1 concluded that the two elements may be the agent's representation that it gathered the HP, and hence that it should now focus on gathering the soul-sphere.**

Then using the action probabilities temporal stacked graph (Fig.8 ②), Expert #1 noticed a specific time interval during which the agent repeated the same action for almost 15 steps. Intrigued by such behavior, Expert #1 replayed this interval and noticed that the agent was within a dead-end (Fig.8 ③) and repeated the action *right* until it changed its orientation up to 180 degrees. Expert #1 commented that observing such interval is interesting because as the agent converges towards an optimal policy, it may have less chances to encounter dead-ends, and thus forgot how to escape them. Expert #1 also observed a similar interval with only *right* actions in which the agent escaped the same dead-end. **Expert #1 concluded that the dead-end was not encoded in the agent's memory, and hence the agent returned to it while searching for items.**

7.3 Feedback from Expert #2

Our second expert, Expert #2, started by re-ordering the memory using the STABLE criteria. He isolated a single hidden state element with continuous activations starting as the agent first saw the HP and remained active until the agent gathered both the red armor and the HP. **Because such element is active regardless of the items the agent gathered yet, Expert #2 interpreted this element as a flag encoding if the agent has seen the health pack or not.**

Then Expert #2 focused on the saliency maps combined with episode playback. He noticed that in one episode, the agent encountered the soul-sphere (last item) before it gathered the red armor (second item). During those time-steps, the saliency maps are not activated towards the soul-sphere despite being the agent's FoV (Fig.8 ⑦), and the memory had no visible changes. Expert #2 intuition was that the agent did not perceived the item. In the final steps of the episode, once the agent gathered the firsts 3 items and then re-encountered the soul-sphere, the saliency maps were activated towards it (Fig.8 ⑧) and the memory activations changed. Expert #2 expressed that *"It is interesting because as soon as it sees it [the soul-sphere] its behavior changes"*. **Expert #2 concluded that the agent intentionally ignored the soul-sphere before it gathered previous items, and as Expert #1 that the agent learned to solve this navigation problem by focusing on one item at a time.**

7.4 Feedback from Expert #3

Expert #3 began his exploration with the t-SNE 2D projection as entry point to identify clusters of hidden states. Expert #3 selected groups of states using the lasso selection (Fig.8 ⑨) to filter the memory timeline. The selected cluster represented consecutive steps, forming a continuous time interval. After replaying this interval, Expert #3 observed that it started at the beginning of the episode and ended when the green armor (first item) entered the agent's FoV. **Expert #3 interpreted this cluster as corresponding to an agent objective, in this case gathering the first item.**

Following up on the previously identified cluster, Expert #3 re-ordered it with the STABLE criteria. Expert #3 noticed one particular hidden state dimension that was activated in blue until the green armor entered the agent's FoV, and then was activated in orange for the rest of the episode. Expert #3 interpreted such element activation as a flag encoding if the agent has seen the green armor. However, after observing this element activations across episodes, Expert #3 noted that it was inactivated (grayish) at the start of an episode. After re-playing this episode he observed that the agent had no armor in its FoV, as opposed to the first episode analyzed where the agent started with the red armor in its FoV. In another episode, where the agent has the green armor in its FoV since the start, the element was constantly activated in orange. **Expert #3 concluded that this element encoded if the agent saw an armor rather than just the green armor.** However, once the agent gathered the green armor, the element remained orange despite still having the red armor in the agent's FoV. **Expert #3 added that this element also encodes if the agent gathered the green armor.**

8 Discussion

In this section, we discuss the collected feedback from experts, as well as the applicability of DRLViz beyond ViZDoom and navigation scenarios.

8.1 Summary of Experts Feedback

Experts filled a post-study questionnaire relative to DRLViz usefulness and usability. Overall DRLViz was positively received by all them: both Expert #1 and Expert #2 stated that DRLViz is *"interesting to explain the behavior of the agent"* and *"easy to use"*. However, Expert #3 stated that he felt *"overwhelmed at first, but soon got used to navigation"*. All 3 experts evaluated the 2D t-SNE projection as the most useful view because it can provide insights on the agent's memory and strategies. They used this view as entry point on at least one episode. They commented that the re-ordering was effective to observe desired hidden states dimensions. Both Expert #2 and Expert #3 used the STABLE criteria because it highlights elements that are different from the rest and should correspond the selected interval. In the other hand, Expert #1 preferred the CHANGE re-ordering criteria because those elements have information concerning the interval. Expert #3 also noted that *"its handy being able to drag it up [derived metrics timeline] and overlay it on the hidden states"* (**G3**).The experts concluded that the agent learned to solve this task sequentially, i. e. by focusing on gathering one item at the time. And thus that the agent only stored information corresponding to which items its has gathered rather than the positions of every seen items at any time-steps.

All 3 experts evaluated the memory reduction interaction that filters the memory view (zoom) not intuitive and hard to use without loosing visual contact with the hidden state dimensions they wanted to focus on. This partially validates our memory reduction goal (**G4**). On this matter, Expert #1 commented that since this agent's memory has 128 dimensions the zoom is not as useful as it could be with 512 dimensions. Expert #2 also commented on the use of the different re-ordering criteria, and that their specific functioning was hard to understand, especially the projection. Expert #3 also mentioned that he *"doesn't fully understand how the projections re-ordering methods are helpful"*. To tackle those issues, Expert #3 suggested to use the derived timeline to re-order the memory, i. e. observe hidden states activations when a feature enters the FoV. Expert #3 also commented that a horizontal zoom could be useful to focus on one particular time interval, and reduce the number of steps to observe. Expert #1 mentioned that brushing the memory while keeping activation areas as *squares*, i. e. both horizontally and vertically could be a better way to implement a more consistent zooming interaction.

8.2 Limits and Perspectives

The memory timeline is the main asset of DRLViz, but also its main limit, as only DRL models with a memory can be used: if a model has no memory, DRLViz is not fully applicable. Another limit is the derived metric timeline that is generated from the simulator that displays information the simulator provides i. e. the items in the agent's FoV. So the current metrics are tied to ViZDoom, and despite being applicable to multiple of its scenarios, if another simulator is used other metrics will be calculated. The essence of our work is applicable to navigation scenarios in robotics and we hope that it could play a role in pushing forward research in intelligent agents (robots) acting autonomously after being trained in simulation. Minor adaptation of the tool to specific environments will be necessary, for instance through the design of metrics. We plan to conduct further research to identify other metrics and extend DRLViz to other simulators mentioned by our experts, such as Matterport3D [26] and Habitat-AI [40] for real-world scenarios, and competitions such as Animal-AI [41] .

DRLViz could be used beyond navigation scenarios, such as death-match [42], however, it may require additional views to highlight the agent's progression towards its goal e. g., killing enemies. Those changes are compatible with the DRLViz interface and workflow, but require additional development to be included. Similar changes would allow to include statistics of the training phase of the model with additional views such as statistics summarizing in [32], and as mentioned by our experts, help them monitor and evaluate the agent during its training.

Scalability is always a concern with visualization techniques. DRLViz supports long episodes and variable memory size. However, if those are larger than the screen real estate (e. g., beyond on average 1200 steps and 800 elements) each memory cell would be smaller than one pixels, and thus difficult to investigate. We plan in the future to support aggregation strategies [43] to provide more compact representation of the timelines. Alignment by event of interest [44] (e. g., gathering HPs) may also provide more compact representations of metrics, and also better support comparison of behavior before and after this event. A concern raised by experts was the communication of insights gained during the exploration process. We plan

TABLE 3
Performances of agents with different memory reduction strategies (each averaged over 100 episodes).

Type of reduction	Steps survived	HP gathered	Poison gathered	Health
Full-memory	**503.98**	37.56	**4.28**	81.47
Half-memory	493.92	**37.88**	4.66	**81.61**
Identified-memory	462.27	34.38	4.91	78.04

to explore summarizing techniques for thoses insights, such as state charts [45] in which each state corresponds to a local strategy e. g., reach an item. Among the many perspectives, *assessing selected memory* is the most promising one we discuss in the next section.

8.3 Perspective: Assessing Selected Memory

We estimated that redundant memory elements as identified in examples (Sec. 6) could be removed without affecting the agent's performance. However, it remains unclear what can be the impact of this reduction on the *global* agent behavior. To quantify the efficiency of such reductions, we implemented a method that allows to generate new episodes in which agents have a limited memory. Technically, we hijack the memory using *mask vectors* with as dimensions than the memory,(made of $0s$ and $1s$) that either completely obstruct the memory element or preserve it based on the memory selections (Fig.1 ④). The selected elements are preserved ($1s$) while the others are obstructed ($0s$). The mask is then multiplied with the current hidden state before each decision. Thus, each obstructed element have values equals to 0, while the others remain untouched.

In order to evaluate the performances of the agent, we generated 100 episodes per memory reduction strategy. During every episodes, we collected the number of steps the agent survived, the number of HPs and PVs the agent gathered, and its averaged health. We evaluated 3 agent with 3 different strategies of memory reduction:

- *Full-memory:* 512 elements from the memory.
- *Half-memory:* 256 top activated elements
- *Identified-memory:* using 12 elements identified in Sec. 6.

Table 3 shows similar performances between agents with full and half memory. One hypothesis to draw is that the agent has at least 256 non-essential elements. We can also observe that the agent with only 12 elements performed less in average with 462 steps survived (as opposed to 504 with full memory), had less HPs gathered and more PVs gathered. DRLViz enables to explore those scenarios and assess at a finer-grained level if a reduction is effective or not. Fig. 9 shows the derived metrics of both the agent with its full memory (Fig. 9 ①) and with its memory reduced (Fig. 9 ②). With its memory reduced, the decision *ambiguity* metric is constantly high. This results in the agent being hesitant and producing mistakes such as running in circles and bumping into walls which could have been avoided using its full memory. This demonstrate the need for more advanced memory reduction strategies to tackle the trade off between the agent's performances and its memory size.

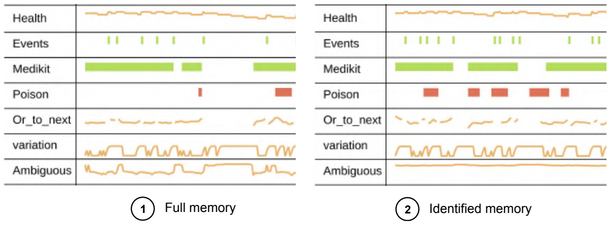

Fig. 9. Timelines of derived metrics based on the agent's activity with full memory ①, and 12 elements identified in scenarios ②. One can observe that the agent using the 12 elements has a constant high ambiguity and a constant high orientation variation.

9 Conclusion

In this work, we introduced DRLViz, a visual analytics interface which allows users to *overview*, *filter* and *select* the memory of Deep Reinforcement Learning (DRL). Analysts using DRLViz were able to explain parts of the memory of agents trained to solve navigation problems of the ViZDoom game simulator, in particular local decisions and higher level strategies. DRLViz received positive feedback from experts familiar with DRL models, who managed to browse an agent memory and form hypothesis on it. DRLViz paves the way for tools to better support memory reductions of such models that tend to be large and mostly inactive. Such reductions need to be validated with further improvements of DRLViz in particular to investigate the role of inactive regions and also complex memory dependencies that are our future research directions.

Acknowledgments

The authors wish to thank our 3 experts Expert #1, Expert #2 and Expert #3 for their valuable time. In particular Expert #1 who shared his model and animated videos.

References

[1] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, p. 269271, 1959.

[2] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, 1968.

[3] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Symposium on Computational Intelligence in Robotics and Automation*, 1997.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2018.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[6] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 10 2017.

[7] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied Question Answering," in *CVPR*, 2018.

[8] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," in *CVPR*. IEEE, 2018.

[9] E. Parisotto and R. Salakhutdinov, "Neural map: Structured memory for deep reinforcement learning," *ICLR*, 2018.

[10] Q. Debard, J. Dibangoye, S. Canu, and C. Wolf, "Learning 3d navigation protocols on touch interfaces with cooperative multi-agent reinforcement learning," in *To appear in European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2019.

[11] F.-Y. Tzeng and K.-L. Ma, "Opening the black box-data driven visualization of neural networks," in *VIS 05. IEEE Visualization, 2005.* IEEE, 2005, pp. 383–390.

[12] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490*, 2016.

[13] F. M. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers," *IEEE Transactions on Visualization and Computer Graphics*, 2019.

[14] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 667–676, 2017.

[15] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning," in *IEEE Conference on Computational Intelligence and Games.* IEEE, 9 2016, pp. 341–348.

[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 11 2017.

[17] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[18] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games*, 2019.

[19] 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.

[20] V. Mnih, A. Puigdomènech Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," Tech. Rep., 2016.

[21] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, 2013.

[22] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 AAAI Fall Symposium Series*, 2015.

[23] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, and others, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[24] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, 2017, pp. 3357–3364.

[25] J. Oh, V. Chockalingam, S. Singh, and H. Lee, "Control of Memory, Active Perception, and Action in Minecraft," 5 2016.

[26] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D Data in Indoor Environments," *International Conference on 3D Vision (3DV)*, 2017.

[27] E. Beeching, C. Wolf, J. Dibangoye, and O. Simonin, "Deep Reinforcement Learning on a Budget: 3D Control and Reasoning Without a Supercomputer," *arXiv preprint arXiv:1904.01806*.

[28] S. Carter, D. Ha, I. Johnson, and C. Olah, "Experiments in Handwriting with a Neural Network," *Distill*, 2016.

[29] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[30] L. Van Der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[31] T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding dqns," in *International Conference on Machine Learning*, 2016, pp. 1899–1908.

[32] J. Wang, L. Gou, H.-W. Shen, and H. Yang, "Dqnviz: A visual analytics approach to understand deep q-networks," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 288–298, 2018.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv:1412.3555 [cs]*, 12 2014.

[34] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 10 2017.

[35] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon, "Visual Analytics: Definition, Process, and Challenges," in *Information Visualization*. Springer Berlin Heidelberg, 2008, pp. 154–175.

[36] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," 12 2014.

[37] A. Light and P. J. Bartlein, "The end of the rainbow? Color schemes for improved data graphics," *Eos, Transactions American Geophysical Union*, vol. 85, no. 40, pp. 385–391, 2004.

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[39] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents," *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[40] 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.

[41] M. Crosby, B. Beyret, and M. Halina, "The animal-ai olympics," *Nature Machine Intelligence*, vol. 1, no. 5, p. 257, 2019.

[42] G. Lample and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[43] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, B. Shneiderman, and M. Taieb-Maimon, *LifeFlow: Visualizing an Overview of Event Sequences*, 2011.

[44] T. David Wang, C. Plaisant, A. J. Quinn, R. Stanchak, B. Shneiderman, and S. Murphy, *Aligning Temporal Data by Sentinel Events: Discovering Patterns in Electronic Health Records*, 2008.

[45] S. Salomón, C. Tîrnăucă, S. Salomón, and C. Tîrnăucă, "Human Activity Recognition through Weighted Finite Automata," *Proceedings*, vol. 2, no. 19, p. 1263, 10 2018.

Theo Jaunet is a PhD Student in Computer Science at INSA de Lyon and LIRIS, a CNRS laboratory (France) since september 2018. He is interested in building visual analytics tools to improve machine learning interpretability and interacting with models. He received his M.S. in Computer Science from université lyon 1 (France) in 2018.

Romain Vuillemot is Assistant Professor of Computer Science at École Centrale de Lyon, Université de Lyon, France. His research interests include information visualization, visual analytics, and human-computer interactions, with a focus on advanced interaction techniques to include humans in the loop of complex predictive analytics processes. Vuillemot has a PhD in computer science from INSA de Lyon.

Christian Wolf is Associate Professor at INSA de Lyon and LIRIS (CNRS), since 2005. He is interested in machine learning and computer vision, especially the visual analysis of complex scenes in motion. His work puts an emphasis on modelling complex interactions of a large amount of variables: deep learning, structured models, and graphical models. He received his MSc in Computer Science from TU Vienna, Austria, in 2000, and a PhD and Habilitation from INSA de Lyon, France, in 2003 and 2012.