DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning

Theo Jaunet, Romain Vuillemot, Christian Wolf

▶ **To cite this version:**

Theo Jaunet, Romain Vuillemot, Christian Wolf. DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning. Computer graphics Forum (Proc. Eurovis), 2020, 10.1111/cgf.13962 . hal-02864138

HAL Id: hal-02864138
https://hal.science/hal-02864138

Submitted on 19 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning

T. Jaunet[1] R. Vuillemot[2] and C. Wolf[1,3]

[1] LIRIS, INSA-Lyon, France
[2]LIRIS, École Centrale-Lyon, France
[3] CITI, INRIA, France

Figure 1: *DRLViz displays a trained agent memory, which is a large temporal vector, as a horizontal heat-map ①. Analysts can* browse *this memory following its temporal construction;* 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 and compare them ④.*

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 due to the number of dimensions, dependencies to past vectors, spatial/temporal correlations, and co-correlation between dimensions. 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 decisions using memory reduction interactions, and to investigate the role of parts of the memory when errors have been made (e.g. wrong direction). We report on DRLViz applied 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 interpretability and explain-ability in the field of visual analytics.

CCS Concepts

• ***Human-centered computing*** → ***Visual analytics;*** • ***Theory of computation*** → *Reinforcement learning;*

1. Introduction

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, social robots and assisting elderly people. Historically, navigation was often solved with discrete optimization algorithms such as Dijkstra [Dij59], A-Star [HNR68], Front-propagation [Yam97] 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 of problems focusing on interactions between an agent and its environment [SB18]. In combination with deep networks as function approximators, this kind of models was very successful in problems like game playing [MKS*15, SSS*17], navigation in simulated environments [DDG*18, GKR*18, PS18], and work in human-computer interaction (HCI) emerging [DDCW19].

The goal of Deep Reinforcement Learning (DRL) is to train agents which interact with an environment. The agent sequentially takes decisions a_t, where t is a time instant, and receives a scalar reward R_t, as well as a new observation o_t. The reward encodes the success of the agent's behavior, but a reward R_t at time t does not necessarily reflect the quality of the agent's action at time t. As an example, if an agent is to steer an autonomous vehicle, receiving a (very) negative reward at some instant because the car is crashed into a wall, this reflects a sequence of actions taking earlier then the last action right before the crash, which is known as the *credit assignment problem*. The reinforcement learning process aims at learning an optimal policy of actions which optimizes the expected accumulated future reward $V_t = \sum_{t'=t}^{t+\tau} R_t$ over a horizon τ.

If agents trained with DRL were deployed to real life scenarios, failures and unexpected behaviors [LCM*18] could lead to severe consequences. This raises new concerns in understanding on what ground models' decisions (e.g., brake) are based [RSG16]. To assess the decision of a trained model, developers [HKPC19] must explore its context (e.g., a pedestrian on the road, speed, previous decisions) and associate it with millions of deep networks parameters which is not feasible manually. Analysing a decision after-the-fact, referred to as post-hoc interpretability [Lip16], has been a common approach in visualization. It consists in collecting any relevant information such as inputs and inner-representations produced while the model outputs decision. With such an approach, DRL experts explore their models without having to modify them and face the trade-off between interpretability and performances. Visual analytics for post-hoc interpretability [HKPC19] yields promising results on tasks such as image classification [OSJ*18], or text prediction [SGPR17]; however, it remains an under-explored challenge for DRL *with* memory.

We built DRLViz, a novel Visual Analytics interface aimed at making Deep Reinforcement Learning models with memory more 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 [HKPC19]). Typically, those experts have already trained agents and want to investigate their decision-making process. We validated DRLViz usability with three experts and report on their findings that informed us on future improvement such as applicability to other scenarios, and novel interactions to reduce the memory of an agent and better find patterns within it.

2. Context and Background

Context and Background

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

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. 2). 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, 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.

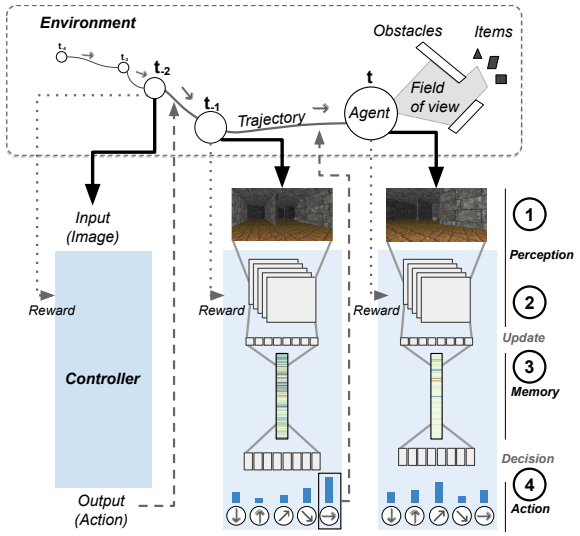

Figure 2: *Our navigation problem consists in solving a visual task (e.g., fetch, interact, or recognize items) while avoiding obstacles in an environment. Deep Reinforcement Learning can be used to solve this problem by using an image as input ① at time t. Features are then extracted from this image 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.*

2.2. Navigation using the ViZDoom Simulation

Navigation using the ViZDoom Simulation

The simulation environment we use to train agents to navigate is ViZDoom [KWR*16] which provides instances of the navigation problem based on Doom, a very popular video game in the 90's. ViZDoom is a 3D world and as such is a proxy problem to mobile service robotics. It supplies different scenarios focusing on various goals (e.g., survive, reach a location, gather items, avoid enemies, etc.). For expert evaluation, and Fig. 2 we used the *k-items* scenario from [BWDS19] with $k = 4$. In this scenario, the agent, walls and items are randomly placed in an environment at the beginning of each episode. Then the agent needs to explore the environment until it succeed, fail or reach a timeout of 525 steps. 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). Gathering an item in the right order grants a $+0.5$ reward r, while failing grants a reward of -0.25. Additionally, the agent receives a reward of -0.0001 at each step. 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 time-step t as follows: $<p_t, (r_t)>$.

This task is challenging as the agent is required to take a decision on the next action based on partial information of the environment, 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 agent is required to recall previously seen observations in some way as it doesn't have access to a global view of the map. These views are stored in its latent memory, the representation studied in this work. The agent should also use its memory to encode information on the items it gathered, and the positions of items or walls encountered in order to quickly complete this task.

2.3. Deep Reinforcement Learning and Memory

Deep Reinforcement Learning and Memory

As expressed in the taxonomy [ADBB17], DRL reached state of the art performance in tasks such as robot control [LFDA16] and board games [SSS*17, JBTR19] where it even surpasses humans in some cases. Recent Deep Reinforcement learning (DRL) models, such as Deep Q-networks (DQN) [MKS*13, MKS*15], and Asynchronous Advantage Actor-Critic (A3C) [MPBM*16], 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 [BNVB13] 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 [HS15] to provide a space the model can use to store an approximation of the history of previous inputs and solve navigation problems [MPV*16, ZMK*17, OCSL16], allowing learning in simulated environments such as Matterport3D [CDF*17], ViZDoom [KWR*16, BWDS19].

2.4. Visual Analytics and Deep Learning

Visual Analytics and Deep Learning

Visual Analytics have been proven to be significantly helpful to deep learning (DL) experts to better understand their models [HKPC19], by providing insights on their decisions and inner representations. Such tools can be applied to Recurrent Neural Networks used as memory. In LSTMVis [SGPR17] users can formulate hypothesis on how the memory behaves with respect to the current input sentence. It displays memory elements in a parallel plot, and by selecting time intervals highlights the most active ones. The re-ordering of memory elements using a 1D t-SNE projection applied to handwriting trajectory prediction [CHJO16] provides an overview of the representation and 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 [KJFF15] 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). RNN evaluator [MCZ*17], uses clustering of memory elements into grids and associate them to word clusters for each input. This tool, also provides additional information on a chosen input in a detail on demand view. RetainVis [KCK*18], a tool applied to the medical domain, studies how a modified model outputs its prediction based on data. With RetainVis, a user can probe an interesting data-point and alter it in a what-if approach to see how it affects predictions. To reach this level of interpretability, the model they used is altered, in a way that reduces its performances, which is different than our approach as visualize the model post-hoc. RNNbow [CPMC], is a tool able to handle different type of input domains, and can be adapted to DRL. However, RNNbow visualize the training of RNNs rather than their decisions. Such a tool, displays the gradients extracted from the model's training, and contextualize it with the input sequence and its corresponding output and label. In DRL with memory, the model does not receive a feedback at each decisions, but rather at the end of the game. This makes RNNbow more difficult implement as it produces large batches on which this tool have issues scaling to. As the authors mentioned, RNNbow targets non-experts user, and a domain specific tool may be required for experts.

As demonstrated by those tools, a decision at a time-step t can be affected by an input seen at t_{-n}. In our case, such inputs are images and experts must first asses what the model did grasp from them before exploring what is stored in the memory. In addition the rewards from navigation tasks, are often sparse which results in a 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). The model interacts with an environment it only sees partially, therefore, its performances can be altered by factors outside its inputs. This forces experts to visualize multiple time-steps in order to analyse a single decision which makes them more difficult to analyse with existing tools.

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 [MPV*16] have been displayed using a t-SNE [VDMH08] 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 [MKS*15], and in [ZBZM16]. DQNViz [WGSY18] displays the training of memory-less models under 4 perspectives. First an overview of the complete training, action distribution of one epoch, a trajectory replay combined with metrics such as rewards and whether an action was random. DQNViz also includes a details view to explore CNN parameters. Such a tool, demonstrates the effectiveness of visual analytics solutions applied to DRL. However, DQNViz focuses on the training of the model, and how random decisions through training can affect it. In addition, the model of DQNViz is limited to fully observable 2D environments in which the only movements available are left or right and thus cannot be applied to navigation tasks. Finally, DQNViz is not designed to display or analyze any memory.

In this paper, we address the under-explored challenge of visualizing a trained DRL model's memory in a 3D partially observed environment. We contextualize this memory with output decisions, inputs, and derived metrics. We also provide interaction to overview, filter, and select parts of such memory based on this context to provide clues on agents decision reasoning and potentially identify how the model uses its memory elements.

3. Model and Design Goals

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

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) [MPBM*16] algorithm. The model is composed of 3 convolutional layers followed by a layer of Gated Recurrent Unit (GRU) [CGCB14], and Fully Connected (FC) layers to match the actions set A. This model is inspired by *LSTM A3C* as presented in [MPV*16] with A3C instead of A2C, and a LSTM [GSK*17] 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. 2 and described as follows:

Stage 1: Environment $\rightarrow$ 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 $\rightarrow$ 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 matrices. 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) $\rightarrow$ New memory. The model maintains and updates a latent representation of its inputs using a Gated Recurrent Unit (GRU) [CGCB14], a variant of recurrent neural networks. This representation, called hidden state h_t, is a time varying vector of 128 dimensions, which is 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 $\rightarrow$ 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.

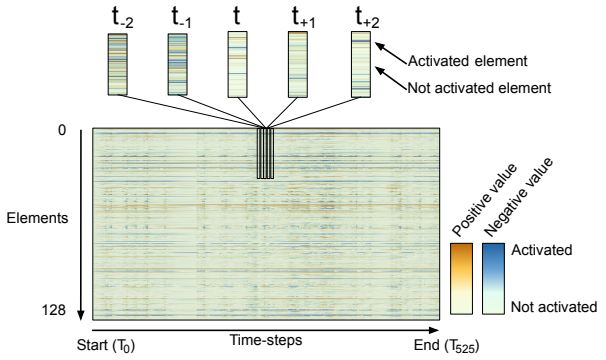

Figure 3: *Memory construction process: at a current time-step t, the agent updates its memory by producing a new memory vector. Each dimension of this vector (represented as a column) is appended to the previous ones chronologically (from left to right). As a result, each row of the appended vectors represent the actions of a single memory element.*

The full set of parameters $\theta = \{\theta_\Phi, \theta_\Psi, \theta_\xi\}$ is trained end-to-end. We used 16 parallel agents and updated the model every 128 steps in the environments. The gamma factor was 0.99, and we used the RMSProp [TH12] optimizer with a learning rate of $7e-4$. We trained the agent over 5M frames, with a frame skip of 4. During training, the agent does not necessary pick the action with the highest probability, as it needs to explore its environment, and eventually find better solutions. However, once the agent is trained, it always chooses the action with the highest probability.

3.2. Constructing the Memory of DRL

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 (128 in our case) time varying vector.

Fig. 3 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. 1). 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 ($h_{t-1} < h_t < h_{t+1}$) 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. 3, 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.

Norms of latent activations are an informative way of visualizing influences [CBYCT19, ZKL*16]. With modern training methods such as weight decay, dropout and batch normalization, it is highly improbable that a high activation can occur for unused features. An alternative to hidden state activations would be to analyze gradients of action probabilities with respect to hidden states [SCD*17, CPMC]. Such an approach can provide information on how a chosen action is directly tied to the current state of the memory, and which dimension influences the more this decision. However, in DRLViz we focus on actions through the episode as a sequence rather than small sub-sequences. When visualizing activations of an LSTM on text, Karpathy et al. [KJFF15] discovered a pattern occurring outside back-propagation limitations of the gradient signal. A solution would be to display both activations and gradients, however preserving the usability and interpratbility of a tool conveying such information is challenge yet to be tackled.

4. Design of DRLViz

Design of DRLViz

We built DRLViz as a visual analytics interface to understand the connections between the latent memory representation (as depicted in Fig. 3) and decisions of an agent trained using Deep Reinforcement Learning. DRLViz primarily exposes the internal memory (Fig. 1) which is interactive and provides *overviewing*, *filtering* and *reduction* both for exploration and knowledge generation [KAF*08]. 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

Design Motivation and Goals

We iteratively built DRLViz with frequent meetings from colleagues experts in DL and DRL (a total of 12 meetings with three experts over 7 months). We first identified their current process to analyze trained agents, e.g., recording videos to playback agents episodes (from its point of view) and decisions (actions probability) to get a sense of the strategy. We also observed experts do a system print of the models' inner values, sometimes add conditions to those prints (e.g., when an item is in the field of view of the agent), and manually look for unusual values. Our approach was to re-build a similar interface in DRLViz with input/output views and facilitate playback, but 1) in an interactive way, and 2) by adding advanced, coordinated views to support advanced models visualization aimed at models developers [HKPC19] (e.g., view on the agent's memory).

Based on a review of the current practices of researchers from our focus group, 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 for navigation problems:

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 built over time (Fig. 3).
G3 Link memory over (a) time and (b) decisions with multiple endpoints, e.g., starting from any time, location, memory or trajectory point.
G4 Filter a sub-set of the memory (a sub-space) tied to a specific agent behavior or strategy.

4.2. Overview and Workflow of DRLViz

Overview and Workflow of DRLViz

Fig. 1 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. 3).
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. 1 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 allwq 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 [SDBR14, GKDF17] representing the segmentation of the image by the agent. The *trajectories* view (Fig. 1) 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 matching the ones on the timeline. The position p_t, and orientation of the agent are represented as an 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* [VDMH08] view of time-steps t using a two-dimensional projection (Fig. 1 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. Dimensions among the selected hidden states can then be re-ordered with any criteria listed in Table 1, and brushed vertically (Fig. 1 ④).

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. 1 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 listed as clickable squares at the bottom left corner of DRLViz.

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.

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.*

Metric	Data Type	Values
Health of the agent	Quantitative	death **[0,100]** full
Event (item gathered)	Flag	**(1)** gathered
Item in FoV	Binary	no item **(0, 1)** item
Orientation to items	Degree	left **[-45,45]** right
Variation of orientation	Quantitative	stable **[0,30]** change
Decision ambiguity	Ratio	sure **[0,1]** unsure

Table 2: *List of* derived metrics *(from top to bottom on Fig. 1 ③)*

4.3. Memory Timeline View

Memory Timeline View

The *memory timeline* exposes the memory's internal structure (**G2**), which is vector (vertical column) of 128 dimensions over 525 time-steps as a heat-map (Fig. 1 ②) from which an interval can be brushed for further analysis. Each cell (square) encodes a quantitative value, whose construction is illustrated in Fig. 3, using a bi-variate color scale from [LB04] 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. 6.4.

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. With the ACTIVATION criteria, a user can observe elements that may be most involved in decisions, while with CHANGE, elements that may be the most used by the model are emphasized, with SIMILAR, a user can see elements with constant activations during selected intervals. In addition of those criteria, we provided the possibility to re-order the memory as presented in [CHJO16] 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.

4.4. Derived Metrics View

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 in finding 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) and outputs (e.g., actions). Finally DRLViz features a stacked area chart of actions probabilities encoding probabilities per action represented by colors corresponding to the ones on the action distribution graph in 1. With this visualization, users can observe similar sequences of decisions.

The derived metrics and stacked area chart 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)) as depicted in Fig. 4. The derived 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**). A constant activation of an element during the same intervals of a metric such as *HP in FoV*, while being different otherwise; may hint that they are related. We provide a full list of 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 three consecutive time-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.

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

Implementation

To explore the memory of a trained agent, one needs to create *instances* of exploration scenarios. For experts evaluations (Sec. 6) we used a trained agent to explore 20 times the environment with different configuration (i.e., positions of items, start position of the agent, its orientation). 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

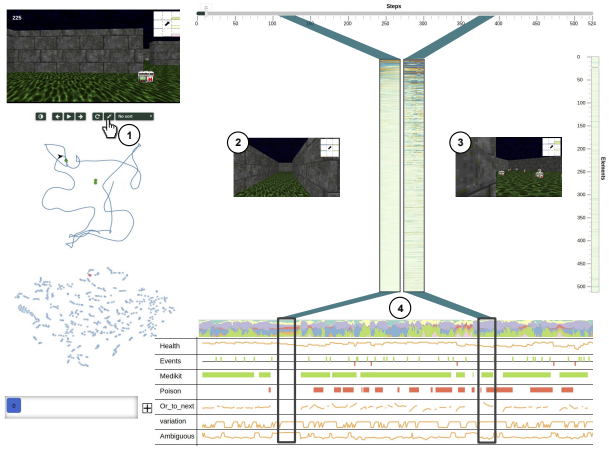

Figure 4: *DRLViz allows to compare selected time intervals ①. For instance to compare when agents face dead-ends ② and when they face health-packs ③. One can observe that more elements are active while the agent is facing HPs than while facing a dead-end. Perhaps those elements are encoding information 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.*

generated using DRL models implemented in Pytorch [PGC*17], and formatted in Python 3. More technical details are provided as supplemental material.

The user interface of DRLViz loads data using JavaScript and D3 [BOH11]. 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 (`https://github.com/sical/drlviz`) and an limited interactive prototype (`https://sical.github.io/drlviz`) are available online.

6. Experts Evaluation

Experts Evaluation

We conducted a user study with three DRL experts who are experienced researchers building DRL models and match the target profile for DRLViz [HKPC19]. We report on their use of DRLViz, as well as insights they gathered. Those results may not be confirmed or denied using DRLViz, but provide hints to formulate hypothesis that can then be studied outside DRLViz e.g., through statistical evidence.

6.1. Protocol and Navigation Problem

Protocol and 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 10 minutes demonstration of DRLViz on a simple ViZDoom scenario: *health gathering supreme*. The evaluation started with DRLViz loaded with data extracted from a model developed by Expert #1, and ended after 35 minutes, during which

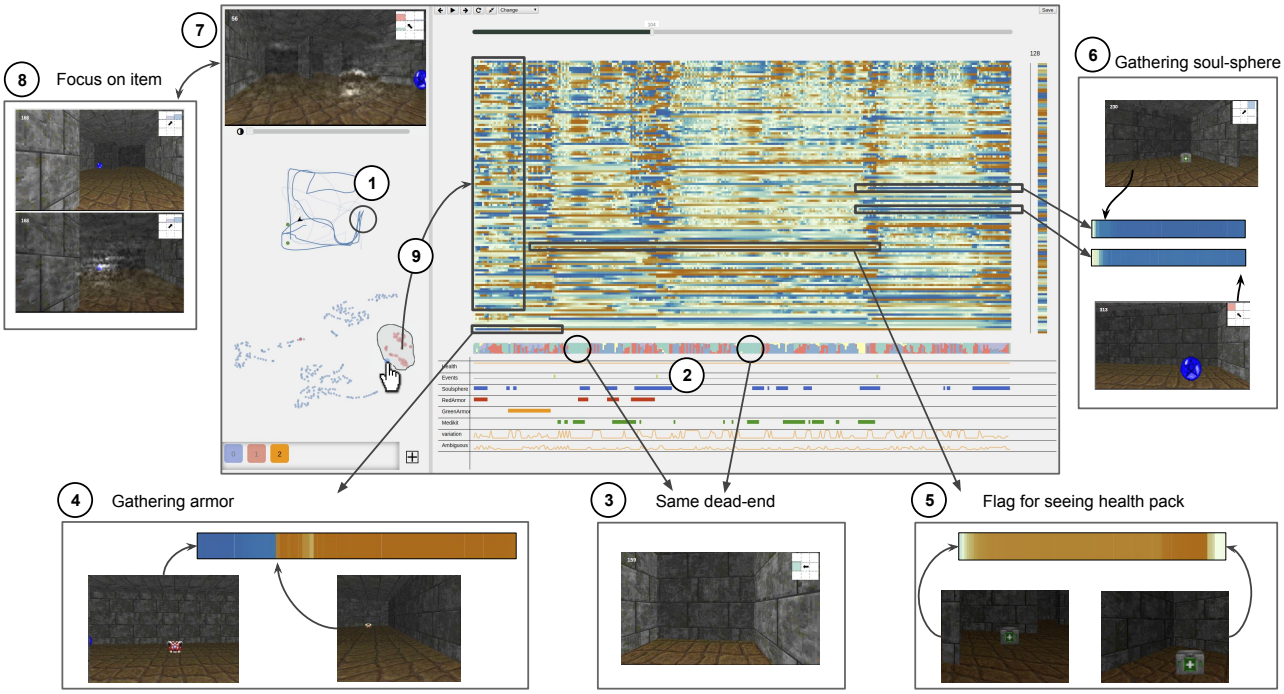

Figure 5: *Summary of the insights gained by the experts. Expert #1 noticed two intervals during which the agent only turned right, by using both trajectory ① and stacked area chart of actions ② views. Once he replayed those sequences, he stated that the agent came twice in the same dead-end ③. Expert #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 he 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 ignored the soul-sphere until it gathered the first three 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.*

experts could explore the displayed data. While using DRLViz experts were told to explain their thoughts and what they wanted to see. Then, experts were asked to fill a post-study questionnaire to collect their first impressions with open questions such as *"Which part of DRLViz was the least useful?"*. The evaluation ended with a discussion guided by the post-study questionnaire on their experience using DRLViz and how it can be improved. The complete evaluation lasted in average 1 hour depending on the length of the discussion. The model used was an A2C [MPBM*16] with 3 convolutional layers and GRU layer with 128 dimensions as memory.

6.2. Feedback from Expert #1

Feedback from Expert #1

Expert #1 is the most experienced expert for this evaluation as he designed both the model and the navigation task [BWDS19] and created animations of agents behaviors. Expert #1 was our primary collaborator to design and build DRLViz.

Fig. 5 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. 5 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 complete the navigation problem by focusing on one item at the time. And 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 area chart (Fig. 5 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. 5 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* ac-

tions 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.**

6.3. Feedback from Expert #2

Feedback from Expert #2

Our second expert, Expert #2, started by re-ordering the memory using the STABLE criteria. He noticed a hidden state element, and zoomed (vertical brush) on it. This element had 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. 5 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. 5 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 mentioned, the agent learned to solve this navigation problem by focusing on one item at a time.**

6.4. Feedback from Expert #3

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. 5 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.**

7. Discussion

Discussion

In this section, we discuss the collected feedback from experts, as well as the limits of the current version of DRLViz.

7.1. Summary of Experts Feedback

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 three 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 on larger memories. 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.

7.2. Limits

Limits

Generalization and *scalability* are the main limits of the current version of DRLViz. Regarding generalization, specific calculations need to be made such as for the derived metrics timeline that is generated from the simulator i.e., the items in the agent's FoV. So the current metrics are tied to ViZDoom but minor adaptation of the tool to specific environments will be needed, but requiring technical knowledge. In the next section we will explain how the interaction techniques in DRLViz can be used beyond the tool for better timeline comparisons. 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 more than 1000 memory dimensions) each memory cell would be smaller than one pixels, and thus difficult to investigate. To tackle such an issue, LSTMVis [SGPR17] introduced a parallel coordinate plot with each line encoding a memory element. However, with DRLViz we sought to support trend detection and thus encode the memory overview using colored stripes [FFM*13] which complies with our data density challenge and requirement to align the memory with the derived metrics below. We then rely on zoom interactions for details for both time and elements.

We plan in the future to support aggregation strategies [WGGP*11] to provide more compact representation of the timelines. Alignment by event of interest [DWPQ*08] (e.g., gathering an item) 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 to explore summarizing techniques for those insights, such as state charts [STST18] in which each state corresponds to a local strategy e.g., reach an item.

8. Perspectives

Perspectives

We present and discuss three works in progress that may be potential improvements of DRLViz, based on experts feedback, that primarily expand its exploration power and generalization.

8.1. Memory Reduction

Memory Reduction

As experts noticed during interviews, agents memory can often be sparse (e.g., Fig.1) or hold redundancy (e.g., Fig. 5 ⑥). Thus we hypothesize that some elements may either never be activated or there might be multiple, redundant activation at the same time. We conducted an experiment to assess that some sub-set of the memory is sufficient to solve a navigation problem, and the rest may be discarded. We used the *health gathering supreme* scenario in which the agent must collect HPs to survive, hence it is easier to solve than k-item. With a larger memory of 512 dimensions, we "removed" hidden state elements by multiplying them by 0 during the experiment.

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**

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

Table 3 shows similar performances between agents with full and top half memory re-ordered with ACTIVATION on the *health gathering supreme* scenario and a large 512 memory of 512 dimensions. One hypothesis to draw is that the agent has at least 256 non-essential elements. Efficient selection of those elements remains a challenge, as it must account complex temporal dependencies. We built an explorable visualization [JVW19] to support this process manually and implemented several strategies, to compare with agents behavior (e.g., being hesitant and producing mistakes such as running in circles or bumping into walls which could have been avoided using its full memory). This paves the way for direct filtering by elements of the memory heat-map in a future version of DRLViz, as the current version only selects temporal intervals.

8.2. Guiding Exploration with Extended Timelines

Guiding Exploration with Extended Timelines

During our interviews, experts suggested to better support the memory analysis process, as the current version of DRLViz relies on visual exploration by the user, with no specific guidance. We identified two areas of improvement for a future version of DRLViz: adding more metrics, and advanced filtering. Regarding the metrics, table 2 introduced derived indicators from the agent decision. Fig.6 illustrate that more metrics can be added using variations of their parameters (e.g., changing variability thresholds or the distance to consider an enemy is in the FoV or not) which support more questions a user may want to investigate. Such metrics are represented in a compact way, and easy to activate by scrolling down, while remaining focused on the memory. Regarding the comparison, the current version only implements *juxtaposition* and *overlay*; while *explicit encoding* [GAW*11] is a third way to compare timelines and memory. We applied this third way by adding a boolean queries builder [LGS*14] using `AND` or `OR` to filter timelines. Those boolean operators are also applicable to all views of DRLViz, such as 2D-map, t-sne or a brush on the memory. This helps users to combine multiple views and answer question such as *Where are the areas of memory with the agent has high health, in this part of the environment, with an enemy and an explosive barrel in FoV?*. This results into intervals in which the agent is susceptible to shoot on barrels to kill enemies.

In order to summarize the input images and ease their comparison with derived metrics, we developed *slit square* interaction based on slit tears [TGF09]. With slit square, a user can brush a square on the inputs. Those squares are then compacted to with the width shared by all time-aligned elements in DRLViz.

8.3. Generalization to other Scenarios and Simulations

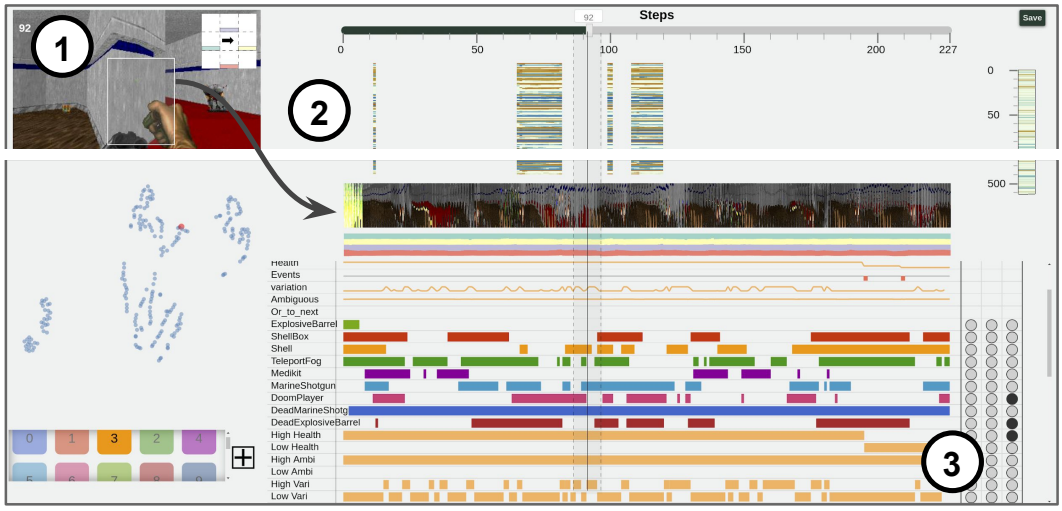

Figure 6: *Extended version of DRLViz loaded on with death-match data. From a slit square selection ① outputs a timeline that summarizes the agent's point of view ②. And the additional metrics and operators ③.*

Generalization to other Scenarios and Simulations

Finally, we started investigating using DRLViz as a general-purpose tool for any trained agents with a memory and spatio-temporal informations. Fig. 6 illustrates DRLViz loaded with a different scenario where the agent shoots towards enemies on the death-match [LC17] with the Arnold model. In general, DRLViz can be used beyond ViZDoom (e.g., referred in [BWDS19]), such as Atari games [WGSY18] without any major change. Using pixel-based representations [Kei00] and zooming [KAL*18] would assure scalability of the timeline representations with scenarios requiring more time steps. We plan to conduct further research to identify other metrics and extend DRLViz to other simulators mentioned by our experts, such as Matterport3D [CDF*17] and Habitat-AI [SKM*19] for real-world scenarios, and competitions such as Animal-AI [CBH19].

9. Conclusion

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.

10. Acknowledgement

Acknowledgement

This research was partially funded by the M2I project http://www.mob2i.fr/, Projet Investissement d'Avenir on urban mobility by the French Environment Agency (ADEME).

References

[ADBB17] ARULKUMARAN K., DEISENROTH M. P., BRUNDAGE M., BHARATH A. A.: Deep Reinforcement Learning: A Brief Survey. *IEEE Signal Processing Magazine 34*, 6 (11 2017), 26–38. doi:10.1109/MSP.2017.2743240. 3

[BNVB13] BELLEMARE M. G., NADDAF Y., VENESS J., BOWLING M.: The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* (2013). 3

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). 7

[BWDS19] BEECHING E., WOLF C., DIBANGOYE J., SIMONIN O.: Deep Reinforcement Learning on a Budget: 3D Control and Reasoning Without a Supercomputer. *arXiv preprint arXiv:1904.01806* (2019). 3, 8, 11

[CBH19] CROSBY M., BEYRET B., HALINA M.: The animal-ai olympics. *Nature Machine Intelligence 1*, 5 (2019), 257. 11

[CBYCT19] CADENE R., BEN-YOUNES H., CORD M., THOME N.: Murel: Multimodal relational reasoning for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 1989–1998. 5

[CDF*17] CHANG A., DAI A., FUNKHOUSER T., HALBER M., NIESSNER M., SAVVA M., SONG S., ZENG A., ZHANG Y.: Matterport3D: Learning from RGB-D Data in Indoor Environments. *International Conference on 3D Vision (3DV)* (2017). 3, 11

[CGCB14] CHUNG J., GULCEHRE C., CHO K., BENGIO Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555 [cs]* (12 2014). 4

[CHJO16] CARTER S., HA D., JOHNSON I., OLAH C.: Experiments in Handwriting with a Neural Network. *Distill* (2016). doi:10.23915/distill.00004. 3, 6

[CPMC] CASHMAN D., PATTERSON G., MOSCA A., CHANG R.: RNNbow: Visualizing Learning via Backpropagation Gradients in Recurrent Neural Networks. 9. 4, 5

[DDCW19] DEBARD Q., DIBANGOYE J., CANU S., WOLF C.: 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). 2

[DDG*18] DAS A., DATTA S., GKIOXARI G., LEE S., PARIKH D., BATRA D.: Embodied Question Answering. In *CVPR* (2018). 2

[Dij59] DIJKSTRA E.: A note on two problems in connexion with graphs. *Numerische mathematik 1* (1959), 269–271. 2

[DWPQ*08] DAVID WANG T., PLAISANT C., QUINN A. J., STANCHAK R., SHNEIDERMAN B., MURPHY S.: *Aligning Temporal Data by Sentinel Events: Discovering Patterns in Electronic Health Records*. 2008. 10

[FFM*13] FUCHS J., FISCHER F., MANSMANN F., BERTINI E., ISENBERG P.: Evaluation of alternative glyph designs for time series data in a small multiple setting. In *Proceedings of the SIGCHI conference on human factors in computing systems* (2013), pp. 3237–3246. 10

[GAW*11] GLEICHER M., ALBERS D., WALKER R., JUSUFI I., HANSEN C. D., ROBERTS J. C.: Visual comparison for information visualization. *Information Visualization 10*, 4 (2011), 289–309. URL: http://ivi.sagepub.com/content/10/4/289.short. 10

[GKDF17] GREYDANUS S., KOUL A., DODGE J., FERN A.: Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138* (2017). 6

[GKR*18] GORDON D., KEMBHAVI A., RASTEGARI M., REDMON J., FOX D., FARHADI A.: Iqa: Visual question answering in interactive environments. In *CVPR* (2018), IEEE. 2

[GSK*17] GREFF K., SRIVASTAVA R. K., KOUTNÍK J., STEUNEBRINK B. R., SCHMIDHUBER J.: LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems 28*, 10 (10 2017), 2222–2232. doi:10.1109/TNNLS.2016.2582924. 4

[HKPC19] HOHMAN F. M., KAHNG M., PIENTA R., CHAU D. H.: Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *IEEE Transactions on Visualization and Computer Graphics* (2019). doi:10.1109/TVCG.2018.2843369. 2, 3, 5, 7

[HNR68] HART P., NILSSON N., RAPHAEL B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics 4* (1968), 100–107. 2

[HS15] HAUSKNECHT M., STONE P.: Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series* (2015). 3

[JBTR19] JUSTESEN N., BONTRAGER P., TOGELIUS J., RISI S.: Deep learning for video game playing. *IEEE Transactions on Games* (2019). 3

[JVW19] JAUNET T., VUILLEMOT R., WOLF C.: What if we reduce the memory of an artificial doom player? URL: https://theo-jaunet.github.io/MemoryReduction/. 10

[KAF*08] KEIM D., ANDRIENKO G., FEKETE J.-D., GÖRG C., KOHLHAMMER J., MELANÇON G.: Visual Analytics: Definition, Process, and Challenges. In *Information Visualization*. Springer Berlin Heidelberg, 2008, pp. 154–175. doi:10.1007/978-3-540-70956-5{_}7. 5

[KAL*18] KERPEDJIEV P., ABDENNUR N., LEKSCHAS F., MCCALLUM C., DINKLA K., STROBELT H., LUBER J. M., OUELLETTE S. B., AZHIR A., KUMAR N., HWANG J., LEE S., ALVER B. H., PFISTER H., MIRNY L. A., PARK P. J., GEHLENBORG N.: Higlass: web-based visual exploration and analysis of genome interaction maps. *Genome Biology 19*, 1 (Aug 2018), 125. URL: https://doi.org/10.1186/s13059-018-1486-1, doi:10.1186/s13059-018-1486-1. 11

[KCK*18] KWON B. C., CHOI M.-J., KIM J. T., CHOI E., KIM Y. B., KWON S., SUN J., CHOO J.: Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE transactions on visualization and computer graphics 25*, 1 (2018), 299–309. 4

[Kei00] KEIM D. A.: Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (Jan. 2000), 59–78. URL: http://dx.doi.org/10.1109/2945.841121, doi:10.1109/2945.841121. 11

[KJFF15] KARPATHY A., JOHNSON J., FEI-FEI L.: Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078* (2015). 3, 5

[KWR*16] KEMPKA M., WYDMUCH M., RUNC G., TOCZEK J., JAŚKOWSKI W.: ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In *IEEE Conference on Computational Intelligence and Games* (9 2016), IEEE, pp. 341–348. 3

[LB04] LIGHT A., BARTLEIN P. J.: The end of the rainbow? Color schemes for improved data graphics. *Eos, Transactions American Geophysical Union 85*, 40 (2004), 385–391. 6

[LC17] LAMPLE G., CHAPLOT D. S.: Playing FPS games with deep reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017). 11

[LCM*18] LEHMAN J., CLUNE J., MISEVIC D., ADAMI C., ALTENBERG L., BEAULIEU J., BENTLEY P. J., BERNARD S., BESLON G., BRYSON D. M., ET AL.: The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv preprint arXiv:1803.03453* (2018). 2

[LFDA16] LEVINE S., FINN C., DARRELL T., ABBEEL P.: End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research 17*, 1 (2016), 1334–1373. 3

[LGS*14] LEX A., GEHLENBORG N., STROBELT H., VUILLEMOT R., PFISTER H.: Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 1983–1992. 10

[Lip16] LIPTON Z. C.: The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016). 2

[MCZ*17] MING Y., CAO S., ZHANG R., LI Z., CHEN Y., SONG Y., QU H.: Understanding Hidden Memories of Recurrent Neural Networks. *arXiv:1710.10777 [cs]* (10 2017). 4

[MKS*13] MNIH V., KAVUKCUOGLU K., SILVER D., GRAVES A., ANTONOGLOU I., WIERSTRA D., RIEDMILLER M.: Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]* (12 2013). 3

[MKS*15] MNIH V., KAVUKCUOGLU K., SILVER D., RUSU A. A., VENESS J., BELLEMARE M. G., GRAVES A., RIEDMILLER M., FIDJELAND A. K., OSTROVSKI G., PETERSEN S., BEATTIE C., SADIK A., ANTONOGLOU I., KING H., KUMARAN D., WIERSTRA D., LEGG S., HASSABIS D.: Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015). 2, 3, 4

[MPBM*16] MNIH V., PUIGDOMÈNECH BADIA A., MIRZA M., GRAVES A., HARLEY T., LILLICRAP T. P., SILVER D., KAVUKCUOGLU K.: *Asynchronous Methods for Deep Reinforcement Learning*. Tech. rep., 2016. 3, 4, 8

[MPV*16] MIROWSKI P., PASCANU R., VIOLA F., SOYER H., BALLARD A. J., BANINO A., DENIL M., GOROSHIN R., SIFRE L., KAVUKCUOGLU K., OTHERS: Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673* (2016). 3, 4

[OCSL16] OH J., CHOCKALINGAM V., SINGH S., LEE H.: Control of Memory, Active Perception, and Action in Minecraft. 3

[OSJ*18] OLAH C., SATYANARAYAN A., JOHNSON I., CARTER S., SCHUBERT L., YE K., MORDVINTSEV A.: The building blocks of interpretability. *Distill* (2018). https://distill.pub/2018/building-blocks. doi:10.23915/distill.00010. 2

[PGC*17] PASZKE A., GROSS S., CHINTALA S., CHANAN G., YANG E., DEVITO Z., LIN Z., DESMAISON A., ANTIGA L., LERER A.: Automatic differentiation in PyTorch. 7

[PS18] PARISOTTO E., SALAKHUTDINOV R.: Neural map: Structured memory for deep reinforcement learning. *ICLR* (2018). 2

[RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), ACM, pp. 1135–1144. 2

[SB18] SUTTON R. S., BARTO A. G.: *Reinforcement learning: An introduction*. 2018. 2

[SCD*17] SELVARAJU R. R., COGSWELL M., DAS A., VEDANTAM R., PARIKH D., BATRA D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 618–626. 5

[SDBR14] SPRINGENBERG J. T., DOSOVITSKIY A., BROX T., RIEDMILLER M.: Striving for Simplicity: The All Convolutional Net. 6

[SGPR17] STROBELT H., GEHRMANN S., PFISTER H., RUSH A. M.: Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics 24*, 1 (2017), 667–676. 2, 3, 10

[SKM*19] SAVVA M., KADIAN A., MAKSYMETS O., ZHAO Y., WIJMANS E., JAIN B., STRAUB J., LIU J., KOLTUN V., MALIK J., PARIKH D., BATRA D.: Habitat: A Platform for Embodied AI Research. 11

[SSS*17] SILVER D., SCHRITTWIESER J., SIMONYAN K., ANTONOGLOU I., HUANG A., GUEZ A., HUBERT T., BAKER L., LAI M., BOLTON A., CHEN Y., LILLICRAP T., HUI F., SIFRE L., VAN DEN DRIESSCHE G., GRAEPEL T., HASSABIS D.: Mastering the game of Go without human knowledge. *Nature 550*, 7676 (10 2017), 354–359. doi:10.1038/nature24270. 2, 3

[STST18] SALOMÓN S., TÎRNĂUCĂ C., SALOMÓN S., TÎRNĂUCĂ C.: Human Activity Recognition through Weighted Finite Automata. *Proceedings 2*, 19 (10 2018), 1263. doi:10.3390/proceedings2191263. 10

[TGF09] TANG A., GREENBERG S., FELS S.: Exploring video streams using slit-tear visualizations. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems* (2009), ACM, pp. 3509–3510. 10

[TH12] TIELEMAN T., HINTON G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning 4*, 2 (2012), 26–31. 5

[VDMH08] VAN DER MAATEN L., HINTON G.: Visualizing Data using t-SNE. *Journal of Machine Learning Research 9* (2008), 2579–2605. 4, 6

[WGGP*11] WONGSUPHASAWAT K., GUERRA GÓMEZ J. A., PLAISANT C., WANG T. D., SHNEIDERMAN B., TAIEB-MAIMON M.: *LifeFlow: Visualizing an Overview of Event Sequences*. 2011. 10

[WGSY18] 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*, 1 (2018), 288–298. 4, 11

[Yam97] YAMAUCHI B.: A frontier-based approach for autonomous exploration. In *Symposium on Computational Intelligence in Robotics and Automation* (1997). 2

[ZBZM16] ZAHAVY T., BEN-ZRIHEM N., MANNOR S.: Graying the black box: Understanding dqns. In *International Conference on Machine Learning* (2016), pp. 1899–1908. 4

[ZKL*16] ZHOU B., KHOSLA A., LAPEDRIZA A., OLIVA A., TORRALBA A.: Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2921–2929. 5

[ZMK*17] ZHU Y., MOTTAGHI R., KOLVE E., LIM J. J., GUPTA A., FEI-FEI L., FARHADI A.: 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. 3