Transparency and Interpretability for Deep Reinforcement Learning: Understanding Decision in Navigation Tasks

Theo Jaunet*
LIRIS, INSA-Lyon

ABSTRACT

Deep reinforcement learning applied to automatic navigation recently achieved significant results using simulators and only images as input in contrast to traditional approaches that often rely on sensors such as lidar. Those learning methods have a wide range of applications such as autonomous cars and social robots. However, they can have unexpected behaviors with severe casualties (e.g. a car crash). To understand a decision, analysts must explore its context and link it with millions of deep network parameters which is not feasible by humans in a reasonable time. As a result, the internal process with which those models make a decision remains misunderstood. This hinders the deployment of these algorithms in real-life situations where regulation and accountability require better interpretability and transparency, to ensure their fairness and safety. With visual analytics tools, analysts can observe how models and their inner parameters behave and thus investigate how it converges towards a decision. We report on the progress of this PhD in this area where we build visual analytics tools, designed to study the memory of those models, and future works aiming at closing the loop on insights discovered to improve future models.

Index Terms: Human-centered computing—Visual analytics—Deep Reinforcement Learning—

1 CONTEXT OF THIS PHD

With the emergence of applications of deep learning to our everyday life (e.g., healthcare), interpreting models' decisions is critical for both end-users to be able to build trust [17] on those models and contest decisions, and for domain experts to assess the fairness and reasoning of those decisions. Among those applications, automatic navigation is one the most challenging problems in Computer Science with a wide range of tasks, 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 [6], A-Star [10], Front-propagation [29], 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 i.e., graph, and are not applicable on tasks that require a visual understanding of a scene (e.g., what are keys). 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 [25]. In combination with deep networks as function approximators, this kind of model is very successfully applied to problems like playing games [19, 22], navigation in simulated environments [4, 8, 20], and work in human-computer interaction (HCI) emerging [5].

This PhD focuses on Deep Reinforcement Learning (DRL) whose goal is to train agents that interact with an environment. As depicted in figure 1, 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.

As trained agents are expected to be deployed to real-world problems in the near future, where failures and unexpected behaviors [16] could lead to severe causalities, their decisions need to be understood. This raises new concerns in understanding on what ground models' decisions (e.g., brake) are based [21]. To assess the decisions of a trained model, developers [11] 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. In addition, analysts must also ensure that their models are not exploiting any bias in their reasoning for, among others, ethical purposes [7] which should also be considered while designing visualizations [3].

2 VISUAL ANALYTICS AND DEEP LEARNING

We started this PhD by focusing on the interpretability of decisions from trained deep reinforcement learning models with memory. The memory of DRL models is a time-varying vector that models updates before each decision based on features extracted from their input (see fig. 1). Analyzing a decision after-the-fact referred to as post-hoc interpretability [17], has been a common approach in the visualization of Deep learning models. It consists in collecting any relevant information such as inputs and inner-representations produced while the model outputs decisions. With such an approach, DRL experts can explore their models without changing them and thus face the trade-off between interpretability and performances. Visual analytics for post-hoc interpretability [11] yields promising results, by providing insights on models' decisions and inner representations. In LSTMVis [24] users can formulate hypotheses 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 [1] provides an overview of the representation and highlight patterns on how different feature dimensions reacts to different path e.g., curvatures.

This PhD approach is related to memory dimensions displayed over the input text of a character level prediction model [14] high-

*e-mail: theo.jaunet@insa-lyon.fr

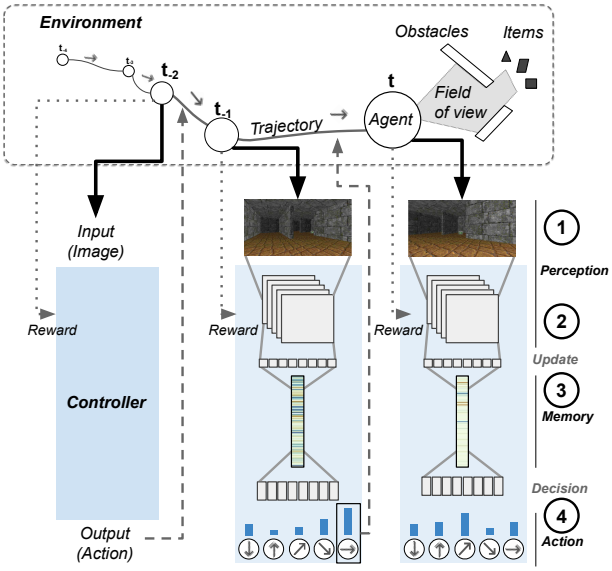

Figure 1: To tackle visual navigation tasks (e.g., fetch, interact, or recognize items) while avoiding obstacles in an environment, Deep Reinforcement Learning can be used. To do so, it uses an image as input ① at time t, features are then extracted from this image ②, and combined with the previous memory vector $t-1$ ③. Using this memory vector, the agent decides on an action such as move forward or turn left, for instance ④.

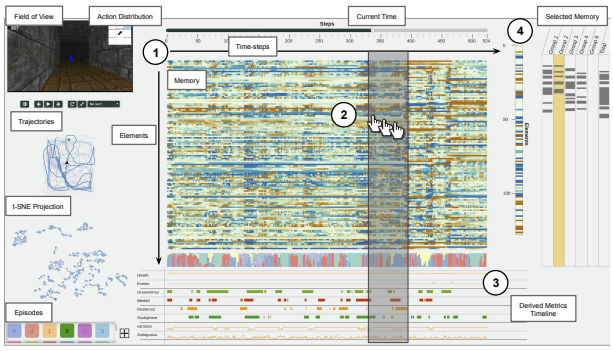

Figure 2: DRLViz displays a trained agent memory, which is a large temporal vector, as a horizontal heat-map ①. Analysts can *browse* this memory following its temporal construction; *filter* according to movements of the agent and derived metrics we calculated ② (e.g., when an item is in the field of view ③); and *select the memory* to filter elements and compare them ④.

lights 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 [18], uses the 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 [15], 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. RNNbow [2], is a tool able to handle different type of input domains, and can be adapted to DRL. However, RNNbow visualizes 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 decision, 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. This PhD expects to follow those previous works while addressing the challenge raised by DRL such as the need to contextualize decisions with information outside models' inputs.

3 Assessing the Memory of DRL Agents

To date, this PhD has focused on the analysis of the memory of Deep Reinforcement Learning agents applied to navigation tasks in simulation.

3.1 Navigation Problem Definition

We focus an a particular navigation problem referred to as k-items. In this problem, an *agent* (e.g., robot, human) moves within a 2D space referred to as *environment* (Fig. 1). An environment contains obstacles (e.g., walls), items the agent may want to gather or avoid, and is bounded (e.g., a room). The goal of the agent is to collect k items in a unique order. 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 a negative reward, finding a certain item may result in a positive reward. To discover and achieve the goal, the agent must explore its environment using actions. 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., a timeout). 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. This work relies on the VIZDoom simulator in which the k-item navigation task is formulated.

As depicted in figure 1, to navigate in a simulation, DRL agents use an image corresponding to its field of view as input and outputs an action. As the agent only uses its field of view as input, each decision is based on a fraction of the environment—i.e., what is in front of it. However, previously seen parts of the environment may be useful for a decision. To tackle this issue, a memory has been implemented in DRL agent. Such a memory consists of a recurrent layer (e.g., LSTM, GRU) which buries features extracted from images into a time-varying vector referred to as a hidden state. This memory changes before each decision and may convey key information on how a decision is made by the agent. In this memory, each row referred to as an element may encode distinct information.

3.2 Contribution 1: DRLViz a Visual Analytics Interface to Understand Decision and Memory [13]

The memory of DRL agents is at the root of their decisions, therefore one hypothesis is that analyzing it may provide insights on their decision process. The accumulation of memory vectors is often represented as heatmaps in which color hues encode intensity which may be interpreted as the corresponding element (i.e., row) having an impact on the final decision. However, due to the size of the memory (128 elements in this case), and the number timesteps that can be done by the agent (i.e., 524 steps until timeout), analyzing such a heatmap is not trivial and can be time-consuming for experts.

We designed DRLViz [13], an interactive visual analytics tool to help DRL experts investigate the memory of their DRL agents and form hypotheses on what ground a decision is based.

In DRLViz, users' explorations of the memory 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

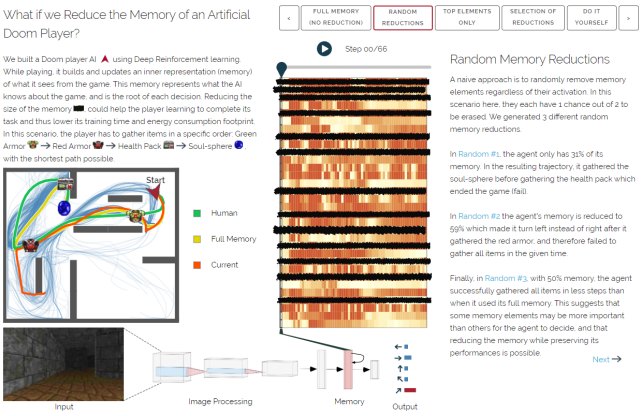

Figure 3: Memory Reduction, an online explorable in which users are invited to try different strategies to reduce the memory of a DRL agent an observe how it affects its behavior.

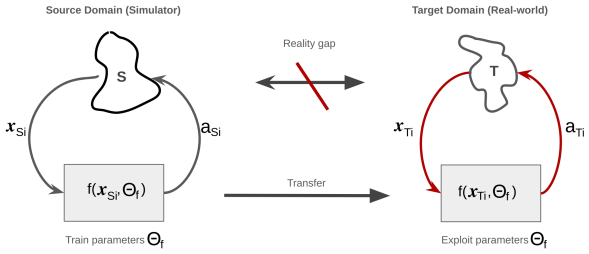

Figure 4: Training an agent in the real-world from scratch requires millions of interactions with its environment and thus is not feasible in a reasonable time. An alternative is to train the agent in a realistic simulation, and then rely on transfer learning approaches to reduce the influence of mismatches between those two environments–referred to as the reality gap).

interface, whose main ones are *image (perception)* and *probabilities (of actions)* which provide context on the agent's decisions. The input image can be animated as a video feed with the playback controls, and a saliency map overlay can be activated [9,23] representing the segmentation of the image by the agent. The *trajectories* view (Fig. 3) displays the sequence of agent positions $p_{t-1} > p_t > p_{t+1}$ on a 2D map. 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. DRLViz, also includes a *t-SNE* [28] 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. Dimensions among the selected hidden states can then be re-ordered based on their activations (e.g., most active, most changing), and brushed vertically (Fig. 3 ④). Both DRLViz, and its code source are available online [1].

3.3 Contribution 2: Understanding the Impact of Memory Reduction in Navigation Performance [12]

The size of the memory (i.e., the number of elements), is a hyperparameter manually set by experts. When deciding on a size, experts often tend to opt for a large memory to ensure that the memory is not a bottleneck preventing the agent from learning complex behaviors. However, one insight discovered by DRL experts while using DRLViz is the redundancy of memory elements. When analyzing how the memory elements encode items the has collected, the experts discovered that some memory elements were having similar activation patterns. In addition, experts also noticed that some remained nearly inactive during episodes. Based on these observations, experts formulated the hypothesis that those elements may be conveying non-necessary information, and thus that they could be removed without affecting the agent's behavior.

To explore such a hypothesis, we designed Memory Reduction [12], an online explorable [2] in which users are invited to test several memory reduction strategies based on elements activations, and observe how the agent behaves with its reduced memory. In this tool, the memory is displayed as a heatmap, and removed elements are represented as black stripes on top of it. Memory reductions are done by multiplying the memory with a mask in which rows to be deleted are zeros, while the rest are ones. This mask is applied to memory before any decision at each timestep, hence removed elements values are forced to remain to zero through episodes. Among reducing strategies such as only keep the top n most activated or most changing elements, the most optimal reduction was achieved with the top quarter of most changing elements and manually selected elements. Those elements were selected based on their activation patterns indicated that they could be related to the encoding of the last item to be collected that the agent had trouble to reach in previous reductions.

[1] `https://sical.github.io/drlviz/`
[2] https://theo-jaunet.github.io/MemoryReduction/

4 Research Agenda

The remaining time of this thesis will be dedicated to deploying DRL agents trained in simulation to real-world environments and robots. Such a deployment will rely on transfer learning approaches like *domain randomization* and *domain adaptation*. Domain experts could benefit from a visual analytics tool designed along with the deployment of those agents. DRLViz will also be improved to provide opportunities to close the loop on insights discovered, and improve the model studied.

4.1 From Simulator to Real Robot

To date, DRLViz has only been used with DRL models in simulators, where ground truth information such as items in the field of view of the agent, as well as the agent's coordinates are provided. Those models trained in simulation cannot be directly deployed into real-life environments and robots as those environments are often different from simulators.

A solution would be to train DRL agents directly in real-life environments, however, as the sample efficiency of DRL is low, agents often require millions of interactions with their environments to grasp their goals and how to achieve them. Such an amount of interaction is not feasible as robots may take several minutes per games as opposed to a simulation that can run thousands of games in seconds. Therefore to tackle such an issue, sim2real transfer approaches are used. It consists in training the agent in a realistic simulation and once the agent behaves well enough, deploy it in a real-world robot. Common approaches to ease such a transfer are domain randomization [26] which focuses on modifying the simulator in order to make the agent more resilient to changes, and domain adaptation [27] which focuses on learning embeddings that suits the agent reasoning through different environments.

However the real-world environment may change, and the simulator may be different from it. This may affect the agent's behavior

once deployed and hinder its performances. For instance, in simulation, we observed that agents tend to exploit limitations, which are not feasible in the real-world (e.g., bumping into walls to move faster). Domain experts need to detect this kind of mistake in the agent behavior, investigate its source, e.g., issue in the physic engine of the simulator, and prevent the agent to exploit it e.g., negative reward on hitting walls. However, those behaviors and their sources are not trivial to investigate and often requires to monitor the agent actions, which can fastidious.

A visual analytics interface could help domain experts to investigate disparities between agents in the simulator and those in the real-world. However, as in real-world environments, ground truth such as the agent coordinates are not available one must rely on other information to align and compare positions and frames to the simulator. An alternative is to train a Deep Learning model to regress the agent's position and orientation from an image corresponding to the field of view of the agent. This can be done using the ground truth coordinates provided by the simulator as a label, and use trajectory smoothing methods to attenuate mistakes that may occur when using such a model in the real-world environment (i.e., reality gap).

With such information, the next step is to analyze the agent deployed in the real-world environment. This can be an entry point for an interactive visual analytics tool that can display gradients of back-propagated agent decisions to identify which parameters and parts of the input (i.e., image in this case) were impactful. This can help experts understand what can be improved in simulation to reduce the reality gap (e.g., dim the lights). Such a tool should also display the agents' inner parameters as with domain adaptation approaches such ADDA [27], which may be relevant to grasp how the agent handles the reality gap in its embeddings to visually assess how effective domain adaptation methods are.

4.2 Supporting Experts in Re-Training their Models

To date, once a hypothesis on memory elements is formulated using DRLViz, experts need to rely on external tools to provide sufficient information to confirm or deny them. Also, once a hypothesis is confirmed, aside from memory reduction, DRLViz offers no opportunities to improve the agent and model. This could be tackled by providing control on the agent's training through direct manipulation. Such manipulations can directly change hyper-parameters and parameters of the agent as it learns. As an example, one could use DRLViz to analyze the memory of an agent through training, and use direct manipulation interactions change the learning rate based on observations. Experts also mentioned that freezing the weights corresponding to memory elements could help the agent to converge towards a behavior (e.g., on memory elements encoding uncommon information such as the last item). If the agent is observed to fail in particular cases, forcing the agent to be on those cases may help it learn how to solve them. For instance, the agent gathering items in the wrong order when both of them are in its field of view. To do so, experts need ways to formulate configurations of the environment (e.g., walls, items and agent positions) for the next training batch. Finally, to date designed tools heavily rely on user exploration which may be fastidious. To reduce such an exploration process, pattern mining methods could be used. Those methods could also help experts detect memory patterns unrelated to metrics or observation and thus long term memory. However, automatic pattern detection may also narrow the exploration of the memory, and experts may miss unexpected behaviors. This raises the challenge of both trusting automatic pattern detection to provide correct information, while preserving exploration opportunities.

5 Questions for the Panel

Given the opportunity to, we would like to discuss with the panel the following points:

How to find the right balance engineering and research? Building visual analytics interfaces applied to deep learning needs heavy programming phases that combine both model-related work and visualization-related work such as data processing and communication between visualization and the model. Combining those two domains takes time, and in most cases cannot be used as a recognized contribution in publications. Hence, during those programming phases, we are often unable to work on scientific articles. What would you recommend to promote such work, and to reduce those phases?

Expert in specific DRL models are a scarce resource, what can be done to evaluate tools? Visual analytics tools for deep reinforcement learning models with memory applied to navigation are highly specific tools. Hence, that narrows down the number of people able to evaluate them. As an example, in DRLViz [13], we managed to reach three experts from our lab that did not take part in the design of the tool to evaluate it qualitatively. Such an evaluation was qualified as a weak point of our contribution in early reviews of DRLViz. What would you recommend to design a stronger evaluation for our future tools? Should we change our focus, and design more accessible tools, and/or with broader applications to domains in order to have more people able to evaluate them despite the risk of having limited use for domain experts?

Finding the balance between visualization and AI? Design visual analytics tools applied to deep learning models requires knowledge on both of those domains. These forms of knowledge gaps can be tackled by discussing with domain experts and collaborators to understand, for instance, what can and should be visualized from DL models, and how such data can be represented. However, those exchanges can be limited by an insufficient understanding of communities. Also, since such tools have contributions in two domains, they may be submitted on both visualization and AI conferences. How can we find a balance between focusing on only one domain, and limit our work by being too dispersed across those domains?

6 Conclusion

As a first step, this thesis was focused on the design of interactive visual analytics tools to help domain experts investigate the memory of Deep Reinforcement Learning agents applied to navigation tasks in simulation. With such a tool experts were able to detect activation patterns and insights on how the agent uses its memory to decide on which action to do. Using such insight experts can reduce the memory to its core elements in order to have less dimension to analyze in further runs. This thesis is now aiming towards deploying agents from simulation to reality using transfer learning approaches and designing visual analytics tools to investigate behavior divergences between an agent in simulation and one in reality.

References

[1] S. Carter, D. Ha, I. Johnson, and C. Olah. Experiments in Handwriting with a Neural Network. *Distill*, 2016. doi: 10.23915/distill.00004

[2] D. Cashman, G. Patterson, A. Mosca, and R. Chang. RNNbow: Visualizing Learning via Backpropagation Gradients in Recurrent Neural Networks. p. 9.

[3] M. Correll. Ethical dimensions of visualization research. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2019.

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

[5] 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.

[6] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1:269–271, 1959.

[7] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *stat*, 1050:2, 2017.

[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] S. Greydanus, A. Koul, J. Dodge, and A. Fern. Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*, 2017.

[10] 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*, 4:100–107, 1968.

[11] 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. doi: 10.1109/TVCG.2018.2843369

[12] T. Jaunet, R. Vuillemot, and C. Wolf. What if we reduce the memory of an artificial doom player? 2019.

[13] T. Jaunet, R. Vuillemot, and C. Wolf. DRLViz: Understanding Decisions and Memory in Deep Reinforcement Learning. *Computer Graphics Forum*, 2020. doi: 10.1111/cgf.13962

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

[15] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE transactions on visualization and computer graphics*, 25(1):299–309, 2018.

[16] J. Lehman, J. Clune, D. Misevic, C. Adami, L. Altenberg, J. Beaulieu, P. J. Bentley, S. Bernard, G. Beslon, D. M. Bryson, 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.

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

[18] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding Hidden Memories of Recurrent Neural Networks. *arXiv:1710.10777 [cs]*, 10 2017.

[19] 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*, 518(7540), 2015.

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

[21] M. T. Ribeiro, S. Singh, and C. Guestrin. 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*, pp. 1135–1144. ACM, 2016.

[22] 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*, 550(7676):354–359, 10 2017. doi: 10.1038/nature24270

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

[24] 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*, 24(1):667–676, 2017.

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

[26] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

[27] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7167–7176, 2017.

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

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