ReLVis: Visual Analytics for Situational Awareness During Reinforcement Learning Experimentation

Emily Saldanha[1,2], Brenda Praggastis[1,2], Todd Billow[1,3], and Dustin Arendt[1,3]

[1]Pacific Northwest National Laboratory,
[2]Data Sciences Group, [3]Visual Analytics Group

Abstract

Reinforcement learning (RL) is a branch of machine learning where an agent learns to maximize reward through trial and error. RL is challenging and data/compute intensive leading practitioners to become overwhelmed and make poor modeling decisions. Our contribution is a Visual Analytics tool designed to help data scientists maintain situation awareness during RL experimentation. Our tool allows users to understand which hyper-parameter values lead to better or worse outcomes, what behaviors are associated with high and low reward, and how behaviors evolve throughout training. We evaluated our tool through three uses cases using state of the art deep RL models demonstrating how our tool leads to RL situation awareness.

CCS Concepts

• ***Human-centered computing*** → ***Visualization systems and tools;*** • ***Computing methodologies*** → *Reinforcement learning; Computational control theory;*

1. Introduction

Reinforcement learning (RL) is a rapidly evolving branch of machine learning that uses trial and error to learn an optimal policy. Models learn the best action given observations based on a feedback signal, e.g. reward function, from the system. RL has significant potential benefit for control systems. For example, an RL algorithm learned how to control the HVAC system in a large building to maintain comfort while minimizing energy [WWZ17]. RL may reduce the need for manual calibration and monitoring by enabling an engineer to specify higher-level goals, allowing the system to more quickly adapt to changes with less human intervention.

However, RL is known to have significant sample inefficiency even compared with other machine learning techniques, e.g., millions of observations and days of training for many commonly used RL benchmark systems [HMvH*17]. Furthermore, as the model gains the necessary experience to learn a good policy, the data scientist iterates over and optimizes different model parameters. So, it is a challenge for the RL practitioner to maintain an understanding of the outcomes of experiments over multiple days and develop an effective understanding of the data generated.

An additional challenge is the evaluation of the behaviors learned by the system. Typically, the performance of an RL model is evaluated using the total reward per task-attempt as a function of training time. However, the reward output is not a robust indicator of whether the behaviors and strategies learned by the system meet the needs and desires of the analyst. In fact, RL models are susceptible to "reward hacking," where the model exploits a flaw in the reward function to achieve a high score without actually learning the desired behavior [AOS*16]. Understanding of the actual behaviors and strategies that are employed by the learned policy is important. This is especially true when experimenting with several different possible reward functions, as the reward output will no longer be directly comparable between runs using different reward functions. Data scientists can become overwhelmed and not understand the performance and learned the model behaviors, which could be improved through better situation awareness during experimentation.

Endsley's definition of Situation Awareness (SA) has three levels [End95], which we map to RL experimentation: **(SA1) perception—**seeing what happened during many iterations of model training; **(SA2) comprehension—**understanding patterns of behaviors across training iterations and over time; and **(SA3) projection—**knowing what would happen under a new set of conditions, such as model parameters or reward function. Good situation awareness should lead to better decision-making, i.e., the selection of more performant and stable models for deployment.

Many approaches to understanding RL models help with SA1 by showing the reward performance as a function of training time [HMvH*17, AWR*17], but this is inadequate to summarize the highly complex behaviors learned by the system. Some RL visualizations oriented towards SA2 have been created, including t-SNE embeddings of observations [JCD*18, ZBM16] and saliency maps [GKDF17]. However, these approaches fail to help the data

DOI: 10.2312/evs.20191168

scientist gain insight without significant *ad hoc* analysis. DQNViz [WGSY19] is a recent tool that partially facilitates SA1&2. However, a limitation of DQNViz is its specialization to a single RL environment having a small, discrete action space.

Our contribution is ReLVis, a Visual Analytics system that directly facilitates SA1&2 during RL experimentation (we leave SA3 entirely to the judgment of the RL practitioner). We address the above challenges by allowing data scientists to understand the relationship between modeling choices and successful learned strategies, explore the relationship between the system's state, the agent's action, and the reward signal, and more deeply understand the patterns of behavior within and between training iterations. Finally, ReLVis is designed for continuous control environments and provides new insight for understanding complex continuous state and action spaces compared to the state of the art.

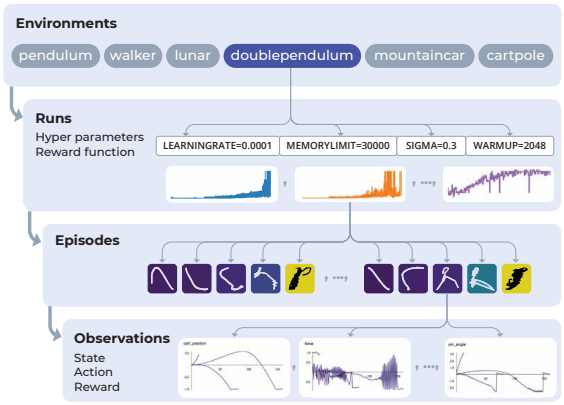

Figure 1: *Reinforcement learning data and terminology*

2. System: ReLVis

ReLVis helps a data scientist maintain SA during RL experimentation by facilitating exploratory data analytics on data logged by RL models. Unlike "traditional machine learning," e.g., supervised classification, RL does not rely on a pre-collected set of training data. Instead, the model learns by taking exploratory *actions* within the *environment* and collecting observational *state* data. The model receives a reward feedback signal designed to guide the agent to a desired behavior or goal state. An RL model uses observations of the environment and the reward signals to make model updates to maximize the cumulative reward. Each attempt that the model makes to accomplish the task at hand is called an *episode*. The model learns how to better accomplish a goal by attempting many episodes within a training *run*.

A data scientist will perform multiple such runs while tuning model hyper-parameters, such as the learning rate or the exploration policy, to optimize the effectiveness of the learning. The relationship between runs, episodes, observations, and other RL terminology is illustrated in Figure 1. ReLVis ingests log files produced during RL experimentation in batches of one or more runs. To produce this log data, we modified the logging functionality of keras-rl [Pla16] to output detailed state, action, and reward logs at each step of training. ReLVis runs analytics on logged data in two phases: when a run is ingested, and also on demand, when a user is interacting with the user interface.

2.1. Data Ingestion & Analytics

A key concept in ReLVis is the "squiggle" which is a thumbnail representation of an episode, found by projecting observations into 2-D. Squiggles are simplified "Time Curves" [BSH*16] intended to provide an impression of the complexity of the behavior within the episode and allow the user to visually assess similarity between episodes. They are used in multiple different views to help the data scientist understand relationships between episodes, reward, and learning. We used Incremental PCA [RLLY08] to ensure squiggles are consistent across runs; the model is partially fitted each time a log file is processed. For environments with periodic behavior, we estimate the spectral density of the observation time series using a periodogram and identify the peak frequency which we take to be the characteristic time scale of the learned behavior. A single representative time slice of this temporal scale was selected from the observations from the midpoint of the time series for each episode to improve the clarity of the squiggle representation. The most recent PCA model and representative time window are stored in a database along with the corresponding runs and episodes.

2.2. Workflow, User Interface, & Visualization

The ReLVis user interface supports the following workflow:

1. **Overview**—the data scientist understands the relationship between hyper-parameters and model performance, and selects runs to explore in detail (Figure 2A). The following views lets the data scientist *perceive* what has happened during experimentation, thus largely supports SA1. The overview is an icicle plot [BN01, KL83] of runs, each containing:
   a. *Reward plot*: the reward of each episode in the run shown as a Sparkline [Tuf06], designed to show the rate and stability of performance improvement; and
   b. *Example Episodes*: the squiggles for five characteristic episodes, designed to show how the complexity of learned behaviors evolves throughout the training process. These are chosen by binning by episode number and selecting the episode with the median reward in each bin.
2. **Discover patterns**—here the data scientist primarily compares runs and episodes at a behavioral level. This supports SA2 by allowing her to *comprehend* patterns and trends across episodes and over time through the following views:
   a. *Squiggle Reward Plot* (Figures 2D, 2F): an arrangement of squiggles where the x-axis is episode number and the y-axis is reward, designed to provide insight into how behaviors evolve according to the selected feedback function;
   b. *Episode Embedding* (Figure 2B): an arrangement of episode squiggles projected with UMAP [MH18], where the distance metric uses the frequency or time domain, designed to show different clusters of behavior or model strategies; and

Environment	# runs	# epis.	# obs.	|A|	|S|
cartpole	70	7,034	690,384	1	4
doublependulum	3	10,963	587,385	1	9
lunar	4	4,588	1,391,966	2	8
mountaincar	141	6,985	4,462,205	1	2
pendulum	16	5,988	1,197,447	1	3
walker	15	15,943	3,766,511	4	24

Table 1: *Runs, episodes, observations, and dimensionality for each environment. $|A|$ and $|S|$ are the action and state dimensions.*

c. *Learning Table* (Figures 2C, 2G): a heatmap designed to show how learned behaviors change during training.

3. **Reveal episode details**—the data scientist compares episodes' raw data. The following views further support SA1:

   a. *Reward and Movie* (Figure 2G): when an episode is selected, the right panel shows that episode's reward over time and a video of the agent's behavior in its environment, this show how the abstract squiggles correspond to real behavior; and
   b. *State & Action* (Figure 2E): an agent's raw state and action functions are shown as line plots, with a separate plot for each dimension, designed to provide additional insights into the actual behavior learned by the system. If multiple episodes are selected, then these are combined into a single plot for easier comparison across episodes.

The squiggle reward plot and episode embedding are scatter plots of episodes. Instead of abstracting episodes as points, we show each episode's squiggle. The size of the squiggle is user-defined, and a greedy algorithm prevents over-plotting by hiding squiggles that would overlap with those already drawn.

3. Evaluation

For evaluation, we apply RL models to standard environments and show via three use cases how ReLVis leads to improved situation awareness. We used Deep Deterministic Policy Gradient (DDPG) models [LHP*15] for continuous tasks and Deep Q Networks (DQN) [MKS*15] for discrete control tasks. To develop and evaluate the visualization capabilities of the tool, we used six control tasks implemented in the OpenAI Gym framework [BCP*16]. Table 1 shows the amount of data generated and the dimensionality of each environment. From these, we selected three use cases of varying difficulty, illustrated in Figure 2, to show how ReLVis led to insights regarding learned strategies and improved situation awareness beyond reward summarization.

Inverted Pendulum (easy): Here the task is to rotate and balance a pendulum in an inverted position—this is a textbook continuous control task. The reward plot view of ReLVis, which allows sorting by reward performance, revealed that some parameter settings led to faster learning (steeper increase in reward) and more stable performance (less dips in the reward towards the end of training). Moving beyond simple reward scoring, we identified three different successful strategies for the task by looking at the squiggle projection view combined with video playback: a single swing to the top, a double swing to the top, and maintaining balance at the top in the cases where pendulum started there. Both the projection layout and the visible similarity of the squiggle shapes allowed us to identify similar behaviors, while viewing the video playback allowed us to identify the corresponding strategy. By observing the squiggles and video playback in the learning table, we found that the double swing strategy was learned earlier in training while the single swing and balancing took more training time to emerge.

Mountain Car (easy): Here the task is to learn to drive a car up a steep hill by building momentum on a neighboring hill. We experimented with a sparse reward, where reward is only provided when the car reaches the goal, and a shaped reward, where the reward function is the car's elevation. Using the detailed reward plot, we observe that the sparse reward leads to more consistent performance results. The squiggles for the shaped reward have longer trajectories indicating that those strategies were more complex. By observing the state and action time series plots showing the direction that the car was moving, we found the sparse reward function leads to strategies that leveraged the neighboring hill. Successful behaviors avoid the temptation of short-term gain of the sub-optimal strategy of immediately climbing the hill.

Bipedal Walker (hard): Here a bipedal robot with four leg joints must learn a walking locomotion to progress forward over uneven ground. This is the most challenging of the six environments. We use the detailed reward plot combined with video playback to demonstrate that while the system was able to learn a successful walking gait, it was not able to maintain the desired behavior in a consistent or stable manner. By looking at the learning table, we find that many episodes demonstrate failed behavior even late in training, with the agent eventually forgetting the best strategies. Using this same view, we found that the type of behaviors exhibited by the agent evolved rapidly, indicating that a lower learning rate might improve the performance of the model.

4. Conclusion

RL is difficult to apply in practice due to sensitivity to hyper-parameters, a time-consuming development process with large volumes of output data, and difficulty evaluating model performance. ReLVis addresses these challenges by providing insights into strategies learned by the model and enabling the user to identify and explain undesirable system behaviors. Future work could support SA3 by helping the data scientist decide what to try next. The techniques we developed for understanding RL data may generalize to other application domains involving collections of time series.

Acknowledgement

The research described in this paper is part of the Control of Complex Systems Initiative at Pacific Northwest National Laboratory. It was conducted under the Laboratory Directed Research and Development Program at PNNL, a multiprogram national laboratory operated by Battelle for the U.S. Department of Energy. We would like to thank Aritra Dasgupta for helpful discussions during the conceptualization of ReLVis.

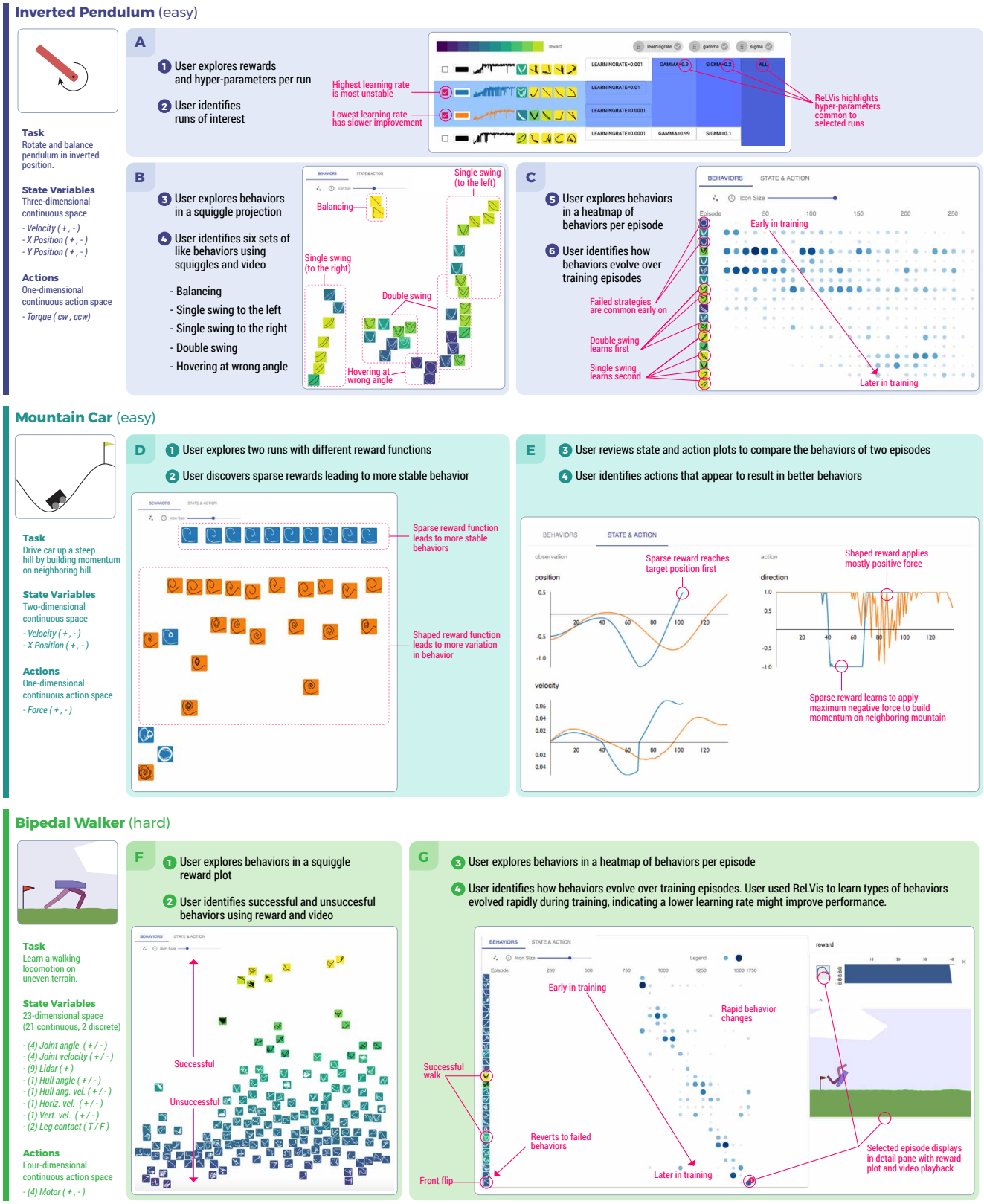

Figure 2: *Use case evaluation of ReLVis across three continuous control environments.*

References

[AOS*16] AMODEI D., OLAH C., STEINHARDT J., CHRISTIANO P. F., SCHULMAN J., MANÉ D.: Concrete problems in AI safety. *CoRR abs/1606.06565* (2016). URL: http://arxiv.org/abs/1606.06565, arXiv:1606.06565. 1

[AWR*17] ANDRYCHOWICZ M., WOLSKI F., RAY A., SCHNEIDER J., FONG R., WELINDER P., MCGREW B., TOBIN J., ABBEEL P., ZAREMBA W.: Hindsight experience replay. *CoRR abs/1707.01495* (2017). URL: http://arxiv.org/abs/1707.01495, arXiv:1707.01495. 1

[BCP*16] BROCKMAN G., CHEUNG V., PETTERSSON L., SCHNEIDER J., SCHULMAN J., TANG J., ZAREMBA W.: Openai gym. *CoRR abs/1606.01540* (2016). URL: http://arxiv.org/abs/1606.01540, arXiv:1606.01540. 3

[BN01] BARLOW T., NEVILLE P.: A comparison of 2-d visualizations of hierarchies. In *infovis* (2001), IEEE, p. 131. 2

[BSH*16] BACH B., SHI C., HEULOT N., MADHYASTHA T., GRABOWSKI T., DRAGICEVIC P.: Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE transactions on visualization and computer graphics 22*, 1 (2016), 559–568. 2

[End95] ENDSLEY M. R.: Toward a theory of situation awareness in dynamic systems. *Human Factors* (1995). 1

[GKDF17] GREYDANUS S., KOUL A., DODGE J., FERN A.: Visualizing and understanding atari agents. *CoRR abs/1711.00138* (2017). URL: http://arxiv.org/abs/1711.00138, arXiv:1711.00138. 1

[HMvH*17] HESSEL M., MODAYIL J., VAN HASSELT H., SCHAUL T., OSTROVSKI G., DABNEY W., HORGAN D., PIOT B., AZAR M. G., SILVER D.: Rainbow: Combining improvements in deep reinforcement learning. *CoRR abs/1710.02298* (2017). URL: http://arxiv.org/abs/1710.02298, arXiv:1710.02298. 1

[JCD*18] JADERBERG M., CZARNECKI W. M., DUNNING I., MARRIS L., LEVER G., CASTAÑEDA A. G., BEATTIE C., RABINOWITZ N. C., MORCOS A. S., RUDERMAN A., SONNERAT N., GREEN T., DEASON L., LEIBO J. Z., SILVER D., HASSABIS D., KAVUKCUOGLU K., GRAEPEL T.: Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *CoRR abs/1807.01281* (2018). URL: http://arxiv.org/abs/1807.01281, arXiv:1807.01281. 1

[KL83] KRUSKAL J. B., LANDWEHR J. M.: Icicle plots: Better displays for hierarchical clustering. *The American Statistician 37*, 2 (1983), 162–168. 2

[LHP*15] LILLICRAP T. P., HUNT J. J., PRITZEL A., HEESS N., EREZ T., TASSA Y., SILVER D., WIERSTRA D.: Continuous control with deep reinforcement learning. *CoRR abs/1509.02971* (2015). URL: http://arxiv.org/abs/1509.02971, arXiv:1509.02971. 3

[MH18] MCINNES L., HEALY J.: Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018). 2

[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 (Feb. 2015), 529–533. URL: http://dx.doi.org/10.1038/nature14236. 3

[Pla16] PLAPPERT M.: keras-rl. https://github.com/keras-rl/keras-rl, 2016. 2

[RLLY08] ROSS D. A., LIM J., LIN R.-S., YANG M.-H.: Incremental learning for robust visual tracking. *International journal of computer vision 77*, 1-3 (2008), 125–141. 2

[Tuf06] TUFTE E. R.: *Beautiful evidence*, vol. 1. Graphics Press Cheshire, CT, 2006. 2

[WGSY19] WANG J., GOU L., SHEN H.-W., YANG H.: Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE transactions on visualization and computer graphics 25*, 1 (2019), 288–298. 2

[WWZ17] WEI T., WANG Y., ZHU Q.: Deep reinforcement learning for building hvac control. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)* (June 2017), pp. 1–6. doi:10.1145/3061639.3062224. 1

[ZBM16] ZAHAVY T., BEN-ZRIHEM N., MANNOR S.: Graying the black box: Understanding dqns. *CoRR abs/1602.02658* (2016). URL: http://arxiv.org/abs/1602.02658, arXiv:1602.02658. 1