Visual Analysis of Action Policy Behavior: A Case Study in Grid-World Driving

Timo P. Gros[1,3,*,†], David Groß[2,†], Julius Kamp[1], Stefan Gumhold[2] and Jörg Hoffmann[1,3]

[1] *Saarland University, Saarland Informatics Campus, 66123 Saarbrücken, Germany*
[2] *TUD Dresden University of Technology, 01062 Dresden, Germany*
[3] *German Research Center for Artificial Intelligence (DFKI), 66123 Saarbrücken, Germany*

Abstract
Learned action policies are gaining ever more traction in AI. One natural idea to support human understanding is visualization of policy behavior, enabling users (domain experts) to visually inspect and scrutinize what has been learned. Here we contribute a case study, in an abstract grid-world driving domain that extends the Racetrack planning benchmark with traffic and with policies that generalize across maps. Our visualization takes as input a set of policy traces, and provides a compact overview of critical situations – crashes or almost-crashes – at map positions, as well as means to explore individual policy traces in depth. Our user study shows that users with access to our visualization obtain a better understanding of critical situations.

Keywords
Visualization, Visual Analysis, Action Policy, Grid-World Driving, ProcGrid Traffic Gym

1. Introduction

Action policies are functions that map states to actions, used as agents in dynamic environments. Learned action policies are gaining ever more traction in AI, in particular neural policies in the context of AI planning where policies need to lead from an initial state to a goal state (e.g. [1, 2]).

However, learned policies are opaque, making it impossible for a human user to understand what has been learned. One natural idea to support human understanding is visualization of policy behavior. Here we address domain experts, whose role is to inspect and scrutinize the policy at system design-time, and either certify the policy for system run-time use, or identify specific shortcomings.

We contribute a case study for this setting, in an abstract grid-world driving domain. Our visualization supports users in understanding policy behavior in critical situations where a crash occurs or almost occurs. Our driving domain substantially extends the so-called Racetrack, a popular benchmark for planning in probabilistic environments (e.g., [3, 4, 5]) where an agent needs to take acceleration actions to navigate from a start line to a goal line without crashing into a wall. We add traffic to this benchmark, with randomly spawned cars driving on the right-hand side of the map's streets. The resulting state space is exponential in the number of traffic participants, much increasing the complexity of Racetrack. We furthermore design neural action policies that generalize across Racetrack maps, with a natural division of labor where a navigation device prescribes the high-level plan (which streets to take), while the learned policy is responsible for accident-free low-level control. While still abstract, our extension thus brings Racetrack much closer to real driving scenarios.

We say that a state is critical if a crash will occur within D steps unless at least one traffic participant changes its velocity. Crashes have $D = 0$, while for $D > 0$ there may still be a possibility to avoid the crash. We provide an overview visualization of the number of $D = 0, 1, 2, 3$ critical situations in each

Late-breaking work, Demos and Doctoral Consortium, colocated with the 3rd World Conference on eXplainable Artificial Intelligence: July 09–11, 2025, Istanbul, Turkey
[*] Corresponding author.
[†] These authors contributed equally.
timo_philipp.gros@dfki.de (T. P. Gros); david.gross1@tu-dresden.de (D. Groß); juka00001@stud.uni-saarland.de (J. Kamp); stefan.gumhold@tu-dresden.de (S. Gumhold); hoffmann@cs.uni-saarland.de (J. Hoffmann)
0000-0002-1100-1952 (T. P. Gros); 0000-0001-8793-9915 (D. Groß); 0000-0003-2467-5734 (S. Gumhold); 0000-0003-1590-5876 (J. Hoffmann)

map position. We provide a detailed visualization for the scrutiny of individual traces, showing the agent action, intended outcome, and actual outcome. While our implementation is specific to our case study, this form of visualization is easy to transfer to any discrete 2D-navigation problem.

In contrast to many XAI works, our visualizations look at the learned policy "from the outside", in terms of the behaviors it induces in the environment. Indeed, while we use neural policies in our case study, our approach is agnostic to the policy representation. Previous work on visualization of policy behavior focused mostly on games. The most closely related prior work is [6, 7], which designs visualizations of statistical model checking results (like crash probability) and Q-values for individual states in the standard Racetrack. We provide a more detailed discussion of related work in Section 5.

We run a user study evaluating our visualization against a simpler baseline, where only the traces and their actual outcomes are shown. The results show that users with access to our visualization obtain a better understanding of critical situations, in terms of the ability to classify given situations correctly; and that they can find specific types of critical situations more quickly.

2. Map-Generalizing Racetrack with Traffic

We now introduce our extended version of Racetrack called *ProcGrid Traffic Gym* (PGTG). Consider Figure 1, which will serve as a running example to explain PGTG.

Environment. Like in Racetrack, the agent's position is defined by its x/y coordinates and velocity. The agent can accelerate or decelerate by one unit per dimension, giving it a choice of nine actions in each state. In Figure 1, the current position is highlighted in blue. The current velocity, indicated by the movement from the previous position to the current one, is $(2, -1)$.

Unlike in Racetrack, the high-level route is obtained by a simple route-finding algorithm, and is visible to the agent in terms of subgoals (shown as light green lines). The agent's task instead is, like in real driving, to navigate local traffic without accidents, in a manner that generalizes across maps (which our policies do).

Grid Cells. PGTG includes several types of grid cells: (i) normal street (white): no additional effect. (ii) ice: may cause the car to slide in a random direction. (iii) sand: may cause the car's tires to spin, reducing velocity to 0. (iv) broken road, which may damage the car's tires, limiting its speed to 1 for the remainder of the episode. Each effect has a customizable probability. Note that each of these effects can apply to any grid cell the agent passes through within a step. For example, in the second-to-last step shown in Figure 1, the agent crossed two broken road cells, facing the possibility of damaging the tires twice.

Traffic. Other traffic participants can enter and exit the grid at any open endpoint. In the running example, there are 9 of these entry points, including the goal (green-black-checkered) and start line (blue). These vehicles follow standard driving rules by remaining in the right lane and make random decisions at intersections and t-crossings. For this study, traffic travels at a constant velocity of 1.

Procedural Map Generation. The map is constructed by combining tiles of size 9×9 grid cells. There are 16 different tiles: straights (2), turns (4), t-crossings (4), crossings (1), dead ends (4) and wall only (1). Our random map generator procedurally generates maps that guarantee to have a connection from start to goal. Map attributes – such as size, tile connectivity, and the amount of sand, ice and broken roads – are fully customizable, allowing for a variety of difficulty levels and challenges.

Additionally, the generation randomly places different kinds of obstacle grid cells around the map.

Training. For this paper, we train a neural agent using Deep Q-Learning [8]. Due to the fixed-size observation window and the repeating structure of pre-composed tiles, the agent can be trained on

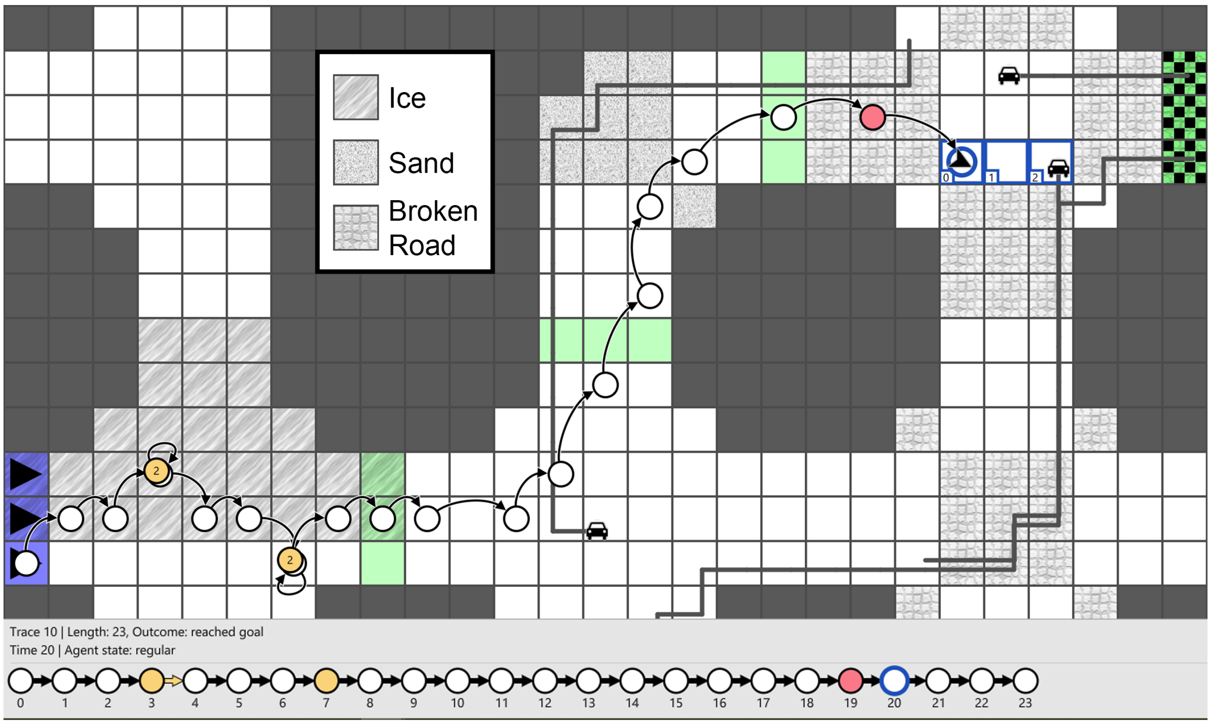


Figure 1: Example of a PGTG environment. The start line is blue, the navigation lines are green and the goal line is green-black-checkered. The current position is highlighted in blue and the discretization of the next step is shown with blue squares. The yellow states are ones where noise occurred, here once on an ice grid cell, and the red one is a critical state. The movement of the cars is indicated by the grey lines.

small maps, such as 3×3 tiles, and later applied to larger maps. For every training episode, a new map is procedurally generated.

This versatile environment presents a challenging learning task. To train effective neural agents, we employ curriculum learning [9], starting with small maps containing only straight cells and no traffic, then progressively adding grid cells of different type and traffic to increase difficulty.

3. Visualization

Our visualization aims to provide means for interactive exploration of a set of traces generated in a PGTG environment. Interaction focuses on states that are deemed critical with respect to possible collisions with obstacles. We therefore provide an overview of such states and a detail view for closer inspection of traces. The visualization is centered around a two-dimensional representation of the environment with square map cells. Start, goal and subgoal cells are colored by their type, while stylized gray-scale images depict obstacles (see Figure 1). Our tool is publicly available.[1]

Critical States. We define the criticality of a state as its time to collision D which is the minimal time until the agent would collide with traffic, given that the velocity of both involved actors does not change and their predicted trajectories intersect [10]. The underlying motion model uses the step discretization scheme as shown in Figure 1. For completeness, we also include direct wall collisions where $D = 0$ by definition. After computing the criticality, all critical states are aggregated by the agent's position and D.

State Overview. As shown in Figure 2, aggregated states are visualized in the physical domain using circular glyphs that are separated into 4 rings. Each of the 4 rings corresponds to a single value of D,

[1]https://doi.org/10.5281/zenodo.15310108

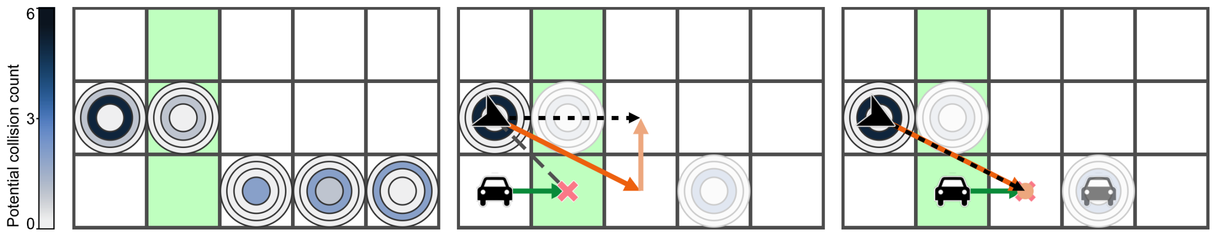


Figure 2: The left image shows a subgoal passage with glyphs of aggregated critical states. The left-most glyph indicates many situations where a crash might occur within $D = 1$ steps. The two right images show a filtered view of instances where the agent could and could not avoid a crash at this position.

with $D = 0$ being in the center. The number of states with the same D is mapped to the color of the respective ring. Lower, and thus more critical, values of D are emphasized by enlarging the inner rings. Selecting a glyph on the map will show a list of all aggregated states on this position. Dotted lines and a red cross indicate the potential collision and and intersection position for the currently selected state.

In addition to the glyphs, a parallel coordinates plot shows for each state its associated criticality and further properties like agent velocity, observed traffic count, wall distance and whether or not this state actually constitutes or results in a collision. The plot aims to show potential dependencies between state properties and outcome. The parallel coordinates plot additionally allows to filter the aggregation to quickly find states of interest. An already filtered example is given in Figure 2 (two right images), where only critical states which may collide in exactly one step are shown.

Trace View. A full trace containing the selected state of interest can be shown using a detailed representation. The agent's path is depicted as a series of circles connected by arrows, each corresponding to a single state and its outgoing transition. A number label shows the count of states located at the same position. Traffic is depicted by gray paths, each with a small random offset to avoid overlapping. A time line (see Figure 1 bottom) allows to select an active state which is then highlighted in blue. Agent and traffic positions are indicated with icons. States are marked yellow if they represent dangerous or irregular scenarios, e.g., when the agent stands still, has damaged tires or experiences noise. Critical states are colored in red and, if selected, the potential collision trajectory is drawn. Examples are shown in Figure 3. Velocity and acceleration of the selected step are shown as orange and green arrows, for agent and traffic respectively. In case of noise, this shows the difference in the agent's actual resulting position and intended target. Optionally, the substeps are shown by marking the visited cells, where number labels state the order. Cells with noise events are shown in yellow borders and crashes in red ones (see Figure 4). These two additions enable a deeper understanding of the state transitions and can help explain why certain situations occur.

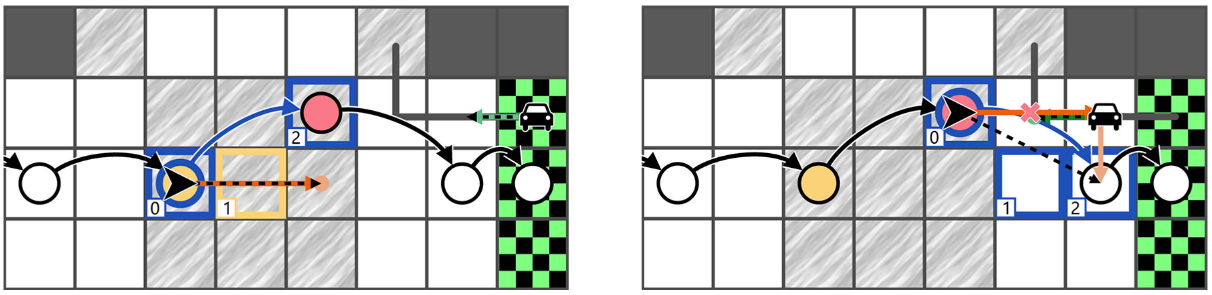


Figure 3: In the left image noise influenced the agent's intended target position. The noise occurred in the tile with the yellow border. The right image shows the resulting state where the agent promptly is in a critical situation, but manages to avoid the oncoming traffic. The arrow and dot in orange indicate the agent's velocity, while the light orange arrow gives its acceleration (zero on the left, to the bottom on the right).

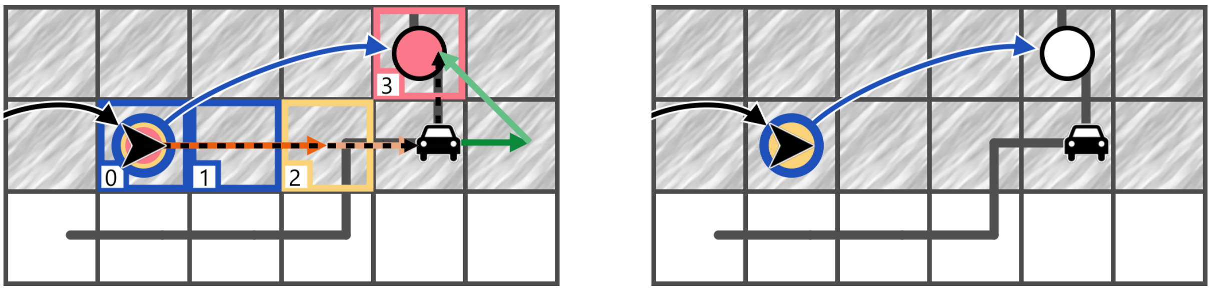


Figure 4: Left image is the baseline and right image our tool. This state is categorized as (ii) critical situation & unavoidable accident. While the baseline group had to infer the agent's velocity from the previous step, the tool group immediately saw that this situation is considered critical and the agent could only accelerate to other critical positions where a crash is inevitable.

4. User Study

In order to measure how well users can understand policy behavior based on our visualization, we conducted a user study. We investigate the users' understanding of the policy based on (i) questions about the policy in specific situations, and (ii) the ability to use the tool for finding specific situations. Hereby, we compare the users' performance when using our tool to a simplified baseline. The study was reviewed and approved without any ethical concerns by the ethical review board of the Saarland university's department of computer science.

Setup. The study participants were students recruited on campus by advertising the study in the computer science building and lectures. Each of the students was compensated with 20 € for participating. The study was conducted in a controlled environment, where each participant had access to a prepared computer with a mouse and keyboard. We had a total of 68 study participants, equally distributed between groups A and B. Group A had full access to our tool as presented in Section 3. The control group B only had access to a baseline consisting of a visualization of traces without showing the state overview or additional details such as velocities, actions and substeps. Critical states were also not highlighted and only states where noise occurred were indicated. Figure 4 shows a comparison of our tool vs. the baseline.

The study consisted of three parts. We started with a warm-up phase where the users had unlimited time and could ask questions in order to learn how to handle the tool. The warm-up included a detailed description of critical situations and two examples, which were also available throughout the complete study. Thereafter, in part 1 of the study, where the time limit was 20 minutes, the users were shown seven specific and distinct situations, and their task was to classify these according to the following scheme: (i) critical situation & avoidable accident, (ii) critical situation & unavoidable accident, (iii) critical situation & accident caused by noise, (iv) critical situation & no accident as prevented through the agent, (v) critical situation & no accident due to luck, (vi) no critical situation & avoidable accident, and (vii) no critical situation & accident caused by noise. For example, the situation in Figure 4 is of type (iii). In part 2 of the study, the users' task was to find situations of interest themselves. Specifically, they had 30 minutes to find one situation for each of the outcome classes (i) – (vii). Additionally, they had to provide an explanation as to why the found situation was of this specific class. The answers were evaluated by two expert users.

Results. Figure 5 presents the outcome of part 1, evaluated separately per question. With the single exception of the first question, group A users answered correctly more often than group B users. The same trend is clearly present in the overall distribution of answers. This indicates the usefulness of our visualization over the baseline. Although the results look promising, when considering the p-values for the six questions (at the bottom of the bars in Figure 5), we see that they did not achieve statistical significance at the 95% level.

For part 2 of our study, since the task was to identify relevant situations within a set time limit, the

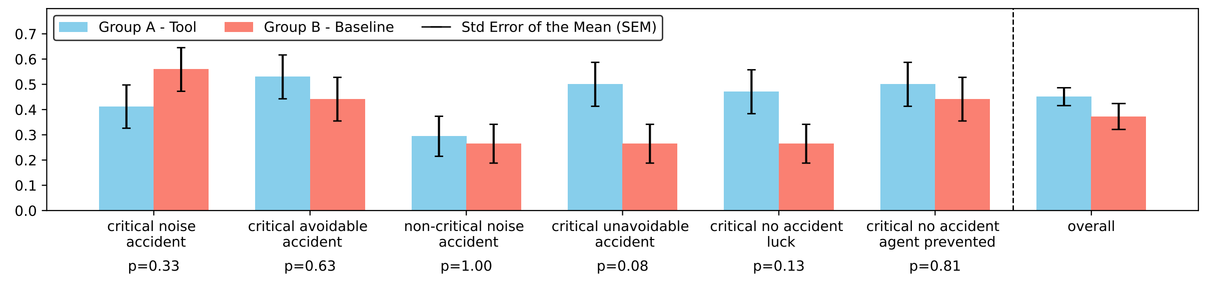


Figure 5: Results of the study's first part as mean percentage of correct answers per question. The provided p-values below the bars were computed with both a Fisher's exact and a chi-square—resulting in the same values—to examine the significance of our results.

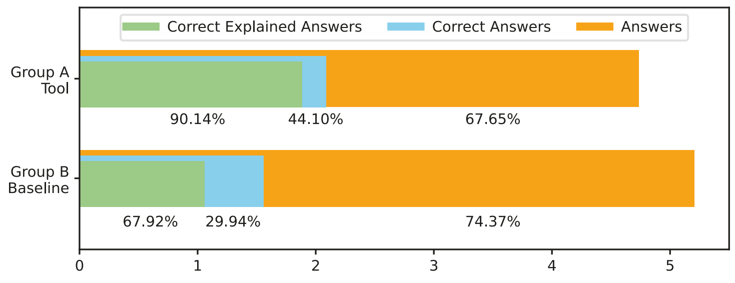
order of findings is irrelevant; only the number of correctly identified situations matters. We report the average number of answers participants provided, the number of correct answers, and the number of correct answers that were also accurately explained.

Consider the bar plot of Figure 6 (left). Our visualization gives group A users a distinct advantage, resulting in approximately 15% more correct answers than the baseline and an impressive 90.14% accuracy in explanations – a 22% improvement over the baseline. These results strongly support that our visualization achieved the desired improvements in explainability. Regarding significance (Figure 6 right), the study reaches statistical significance at the 99% level. We examined the effect by using Cohen's d. These values certify a medium effect for the number of *Correct Answers* and even a large effect for the number of *Correct Explained Answers*.

We remark that, in the above, we did not filter the participants at all, i.e., all users are included regardless of some odd answers (like "the driver was asleep"). We experimented with several variants of user filtering. None of these changed the results, i.e., our findings are robust to user filtering.

5. Related Work

Related Visualization Approaches. The most closely related prior work is [6, 7], which designs visualizations of statistical model checking results (like crash probability) and Q-values for individual states. This pertains however to the structure of standard Racetrack states, consisting exclusively of agent position and velocity; it is very unclear how to extend it to deal with traffic. In a similar vein to our approach, [11] use the notion of critical states to allow judging the trustworthiness of a policy by end users. Critical states defined by very high or low reward scenarios have been visualized for Atari games in human interpretable ways and used to judge the situational awareness of the system [12]. Other approaches build on more abstract views, like saliency maps, state clustering methods like t-SNE, or attention heat maps and are typically explored for policies playing Atari games or Go [13, 14, 15].



	Significance p-value	Effect
Correct Answer	0.0098 (χ^2) 0.0092 (Fisher's exact)	0.30 (Cohen's d)
Cor. Expl. Answer	0.004 (χ^2) 0.0026 (Fisher's exact)	0.58 (Cohen's d)

Figure 6: Results of part 2 for both groups. The bars (left) depict the absolute values for the number of given *Answers*, *Correct Answers*, and *Correct Explained Answers*. We additionally provide the relative values below the bars. The relative value of *Answers* is defined with respect to the total number of questions (7), the relative value of *Correct Answers* with respect to the number of given *Answers*, and the relative value of *Correct Explained Answers* with respect to the number of *Correct Answers*. The table (right) reports p-values (computed by both χ^2 and Fisher's exact) and to assess the significance of the results and Cohen's d-values to quantify the effect size.

Related Benchmarks. Procedurally generated benchmarks have been used prominently, as they offer a wide range of different scenarios and can be used to test the generalization capabilities of a policy. To name just a few, OpenAI's Procgen [16] provides a suite of procedurally generated 2D computer games. Avalon [17] or Obstacle Tower [18] are both 3D benchmarks with a discrete action space providing a variety of different tasks with increasing difficulty. MetaDriverse [19] is the most related to PGTG, as it also generates a road network from different pre-composed building blocks. However, MetaDriverse constitutes a continuous control task in an autonomous driving environment, while PGTG is a discrete control task in a grid-world environment.

6. Conclusion

Supporting humans in understanding neural action policies undoubtedly is an important challenge in AI today. Our work contributes a step in this direction, through a new benchmark for case studies of navigation in traffic, through a first visualization tool, and through empirical evidence that this visualization can be useful.

One interesting topic for future work is the integration of deeper analyses and analysis result visualization, such as statistical model checking results and Q-values in prior work on Racetrack [6, 7], or policy testing methods analyzing avoidability of failures [20, 21]. Methods will need to be found here for filtering among the exponentially many traffic situations in a given map region. We also believe that our ideas can be applied to continous rather than discrete navigation. In the longer term, a question is how human input, based on visualization-supported human understanding, can be integrated in the policy training process. This could be done via a visual interface allowing the user to specify behavior constraints (e.g. maximum speed within a user-defined region), and then revising the policy according subject to these constraints, providing a strong form of quality assurance.

Acknowledgments

This work was partially supported by the German Research Foundation (DFG) under grant No. 389792660, as part of TRR 248, see https://perspicuous-computing.science, by the German Federal Ministry of Education and Research (BMBF) as part of the project MAC-MERLin (Grant Agreement No. 01IW24007), by the European Regional Development Fund (ERDF) and the Saarland within the scope of (To)CERTAIN, and by the German Federal Ministry of Education and Research (BMBF, SCADS22B) and the Saxon State Ministry for Science, Culture and Tourism (SMWK) by funding the competence center for Big Data and AI "ScaDS.AI Dresden/Leipzig". We further thank E. Wuenscher, I. Hoese, and N. Waltner for supervising the participants during the study.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT 4o for grammar and spell checking. Afterwards, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Toyer, S. Thiébaux, F. W. Trevizan, L. Xie, Asnets: Deep learning for generalised planning, Journal of Artificial Intelligence Research 68 (2020) 1–68.

[2] S. Ståhlberg, B. Bonet, H. Geffner, Learning general optimal policies with graph neural networks: Expressive power, transparency, and limits, in: Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22), AAAI Press, 2022, pp. 629–637.

[3] A. G. Barto, S. J. Bradtke, S. P. Singh, Learning to act using real-time dynamic programming, Artificial Intelligence 72 (1995) 81–138.

[4] B. Bonet, H. Geffner, Labeled RTDP: Improving the convergence of real-time dynamic programming, in: Proc. ICAPS'03, 2003, pp. 12–21.

[5] T. P. Gros, N. J. Müller, D. Höller, V. Wolf, Safe reinforcement learning through regret and state restorations in evaluation stages, in: Principles of Verification: Cycling the Probabilistic Landscape: Essays Dedicated to Joost-Pieter Katoen on the Occasion of His 60th Birthday, Part III, Springer, 2024, pp. 18–38.

[6] T. P. Gros, D. Groß, S. Gumhold, J. Hoffmann, M. Klauck, M. Steinmetz, TraceVis: Towards visualization for deep statistical model checking, in: Proceedings of the 9th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'20), 2020.

[7] D. Groß, M. Klauck, T. P. Gros, M. Steinmetz, J. Hoffmann, S. Gumhold, Glyph-based visual analysis of q-leaning based action policy ensembles on racetrack, in: 26th International Conference Information Visualisation (IV'22), IEEE, 2022, pp. 1–10. doi:10.1109/IV56949.2022.00011.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (2015) 529–533.

[9] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

[10] L. Westhofen, C. Neurohr, T. Koopmann, M. Butz, B. Schütt, F. Utesch, B. Neurohr, C. Gutenkunst, E. Böde, Criticality metrics for automated driving: A review and suitability analysis of the state of the art, Archives of Computational Methods in Engineering 30 (2023) 1–35.

[11] S. H. Huang, K. Bhatia, P. Abbeel, A. D. Dragan, Establishing appropriate trust via critical states, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3929–3936.

[12] C. Rupprecht, C. Ibrahim, C. J. Pal, Finding and visualizing weaknesses of deep reinforcement learning agents (2019). URL: http://arxiv.org/abs/1904.01318. arXiv:1904.01318.

[13] T. Zahavy, N. B. Zrihem, S. Mannor, Graying the black box: understanding dqns, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML'16, JMLR.org, 2016, p. 1899–1908.

[14] J. Wang, L. Gou, H.-W. Shen, H. Yang, DQNViz: A visual analytics approach to understand deep q-networks, IEEE Transactions on Visualization and Computer Graphics 25 (2019) 288–298.

[15] S. Greydanus, A. Koul, J. Dodge, A. Fern, Visualizing and understanding Atari agents, in: International Conference on Machine Learning, PMLR, 2018, pp. 1792–1801.

[16] K. Cobbe, C. Hesse, J. Hilton, J. Schulman, Leveraging procedural generation to benchmark reinforcement learning, arXiv preprint arXiv:1912.01588 (2019).

[17] J. Albrecht, A. Fetterman, B. Fogelman, E. Kitanidis, B. Wróblewski, N. Seo, M. Rosenthal, M. Knutins, Z. Polizzi, J. Simon, et al., Avalon: A benchmark for rl generalization using procedurally generated worlds, Advances in Neural Information Processing Systems 35 (2022) 12813–12825.

[18] A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, D. Lange, Obstacle tower: A generalization challenge in vision, control, and planning, arXiv preprint arXiv:1902.01378 (2019).

[19] Q. Li, Z. Peng, L. Feng, Q. Zhang, Z. Xue, B. Zhou, Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning, IEEE transactions on pattern analysis and machine intelligence 45 (2022) 3461–3475.

[20] H. F. Enişer, T. Gros, V. Wüstholz, J. Hoffmann, M. Christakis, Metamorphic relations via relaxations: An approach to obtain oracles for action-policy testing, in: ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'22), 2022.

[21] M. Steinmetz, D. Fišer, H. F. Enişer, P. Ferber, T. Gros, P. Heim, D. Höller, X. Schuler, V. Wüstholz, M. Christakis, J. Hoffmann, Debugging a policy: Automatic action-policy testing in ai planning, in: Proceedings of the 32nd International Conference on Automated Planning and Scheduling (ICAPS'22), AAAI Press, 2022, pp. 353–361.