Glyph-Based Visual Analysis of Q-Learning Based Action Policy Ensembles on Racetrack

D. Groß*, M. Klauck†, T. P. Gros†, M. Steinmetz†, J. Hoffmann†, S. Gumhold*
*TU Dresden, Germany. *david.gross1—stefan.gumhold@tu-dresden.de*
†Saarland University, Germany. *klauck@depend.uni-saarland.de,timopgros—steinmetz—hoffmann@cs.uni-saarland.de*

***Abstract*—Recently, deep reinforcement learning has become very successful in making complex decisions, achieving super-human performance in Go, chess, and challenging video games. When applied to safety-critical applications, however, like the control of cyber-physical systems with a learned action policy, the need for certification arises. To empower domain experts to decide whether to trust a learned action policy, we propose visualization methods for a detailed assessment of action policies implemented as neural networks trained with Q-learning.We propose a highly responsive visual analysis tool that fosters efficient analysis of Q-learning based action policies over the complete state space of the system, which is essential for verification and gaining detailed insights on policy quality. For efficient visual inspection of the per-action Q-value rating over the state space, we designed three glyphs that provide different levels of detail. In particular, we introduce the two-dimensional Q-Glyph that visually encodes Q-values in a compact manner while preserving directional information of the actions. Placing glyphs in ordered stacks allows for simultaneous inspection of policy ensembles, that for example result from Q-learning meta parameter studies. Further analysis of the policy is supported by enabling inspection of individual traces generated from a chosen start state. A user study was conducted to evaluate the effectiveness of our tool applied to the Racetrack case study, which is a commonly used benchmark in the AI community abstracting driving control.**

***Index Terms*—Glyph visualization, visual analysis, neural networks, action policy verification**

I. Introduction

Machine learning has undergone an uplift in interest over the past few years. The idea of using a neural network (NN) to replace traditional approaches, that may require sophisticated algorithms or are impracticable to solve otherwise, has been successfully applied to many areas of computer science, such as image classification [1], natural language processing [2], and game playing [3]–[5]. In particular, NNs are the technical core of ever more *intelligent systems*, created to assist or replace humans in decision-making, in important applications such as industry automation, trading, and autonomous driving. In such intelligent systems, the NN is used as an *action policy*, a function that maps system and environment states to actions, thus controlling the system behavior. This development comes with the urgent need for *certification* of NN action policies. If neural networks are in control of decisions affecting economic value or even human lives, then engineers need to be able to assess their behavior and confirm that desirable behavioral properties like safety are satisfied – or find that this is not the case and identify specific weaknesses for re-training in reinforcement learning. We argue that NN action policy visualization can and should become a core tool for such certification. We need to enable engineers to inspect the behavior of the learned policy, and effectively understand what has been learned and where potential weaknesses are. Such visualization can go hand-in-hand with system analyses such as formal verification, where applicable. Indeed, previous work [6] has introduced the TraceVis Tool which visualizes statistical model checking results [7] as an overview of NN policy behavior across the state space.

Such NN policies can be trained in a multitude of ways. We here consider *deep Q-learning* [8], where the NN controller is iteratively trained by interacting with the environment and updating the policy. In each step, the current policy is executed, and its NN weights are updated using gradient descent. The NNs are used to approximate per action the expected discounted reward, the so-called Q-value, that is accumulated when deciding for action a and following the policy afterwards. An arg-max function on the Q-values is used to determine the next action. The so called *discount factor* γ with values ranging between 0 and 1 weights the importance of short-term to long-term rewards. Deep Q-learning has been used successfully on several decision-making problems [3], [4], [8]. Moreover, it has explicitly been shown to perform better than other methods on the Racetrack benchmark [9], which we use as a test case.

In this work, we extend the TraceVis approach by visual analysis of the NN representing the action policy, and focus our analysis on the output layer that provides the Q-values. We present a highly interactive visualization tool that enables the user to identify and gain insight over state space regions where the Q-values exhibit unexpected, dangerous, or only-just-safe behavior. Key to accomplish this is an overview visualization of the system state space, and key to such an overview is a compact visualization of Q-values in each state.

To this end, we designed three different glyph-based visualizations with different levels of detail, to enable efficient visual inspection of the network decisions as shown on left side of Figure 1. We do so in the context where the action policy takes decisions pertaining to movement in 2D, i.e., in a plane. Such movement decisions are a core aspect of any cyber-physical system that moves autonomously on ground (driving, factory halls, etc). In this setting, a key feature of actions is direction in a plane. Our visualization targets and leverages that feature. Our first glyph variant shows the chosen action for each state and acts as an overview visualization. The

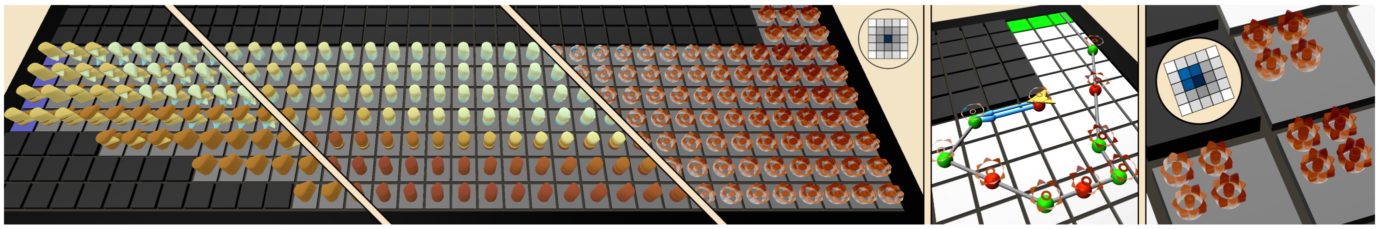
Fig. 1. Example visualizations of a policy ensemble using our three glyph types *Chosen Action*, *Directional Aggregation* and *Q-Glyphs* (left), a single trace reaching the goal under strong influence of noise (middle) and placement of glyphs for multiple velocities near a corner position on *Barto-small* (right).

second variant aggregates over all action directions providing a visual impression of the Q-value distribution over the actions. A deeper look is enabled by visualizing the individual Q-values using our third glyph type, which we call the Q-Glyph. Finally, full traces can be generated from a given start state as illustrated in the middle of Figure 1.

We evaluate the prototype by means of a user study, where participants are asked to inspect a given policy for its safety, plausibility and possible weak spots. We furthermore conduct a separate case study, using our tool to compare different policy stages during reinforcement learning, as well analyzing the impact of different hyper-parameter settings.

In summary, our main contributions are:

- Q-Glyphs, a visual encoding for Q-values which relate to the direction of movement-related actions in a plane.
- A visual analysis approach to efficiently inspect action policies using overview and detail visualizations.
- A study of the proposed approaches for Q-learning based action policies in the Racetrack benchmark, including a user study, and an evaluation of policy ensembles on several scenarios in a case study.

II. Related Work

We next discuss related work on certification of NN-based action policies, visualization of such action policies, glyph-based and ensemble visualizations.

Formal Action Policy Verification: The NNs we consider represent action *policies* taking decisions in (exponentially) large probabilistic transition systems (aka state spaces), specifically Markov decision processes (MDP) [10]. Given a fixed action policy, the induced behavior in the MDP is a Markov chain. Full formal verification of NN action policy behavior on all states is exceedingly hard, due to state space size ("state space explosion problem") in combination with the complexity of NN analysis. Most current work is limited to single NN decisions [11]–[15], as opposed to sequences of NN decisions in a large state space. Recent work proposed to analyze such decision sequences, through statistical model checking over the underlying Markov chain [7], [16], [17]. This approach is limited however to the analysis of individual start states from which the policy can be run. Racetrack, a benchmark from AI autonomous decision making, has recently been taken up in the verification community [18] as a manageable abstraction of autonomous driving, providing a simple yet meaningful and highly extensible research environment. We adopt Racetrack and detail it in Section III. Tools related to Racetrack can be found on https://racetrack.perspicuous-computing.science/.

Visualizing NN Action Policies: Hohman et al. [19] present a broad survey of general methods that have been used for visualization of NN learning and behavior analysis. One of the more prominent examples of visualization in NN analysis is its application to policies playing Atari 2600 games. In this context, Wang et al. [20] developed DQNViz, a visual analysis tool for inspection of deep NNs. They use a comprehensive collection of views showing training and epoch statistics, like average reward, loss and action distributions, and use a trajectory view to examine movement and reward patterns. An image-based approach is presented by Greydanus et al. [21], who use saliency maps to analyze the agents strategy to maximize its reward and judge the quality of its actions. In a similar fashion, Joo et al. [22] make use of Grad-CAM [23] to foster the understanding of the hidden layers of a CNN action policy. The Grad-CAM approach was also applied to the Go policy network [3] in a work by Pang and Ito [24]. A visual representation of the game board visualizes model features using color and opacity, with the goal to provide explanations for the next movements of the agent. Gros et al. [6] developed the TraceVis tool to visualize the outcome of statistical model checking of action policies. The approach aims at providing an overview of model checking results obtained in individual start states, such as the probability of reaching the goal, allowing to interactively delve into policy behavior, down to individual policy runs. Hämäläinen et al. [25] propose a three-dimensional height field visualization, based on a previous work by Li et al. [26], to help with the optimization of movement control in the context of humanoid locomotion. While their focus is not on action policies, it shows how such visualizations can be used to optimize NNs.

Glyph-based Visualization: The usage of glyphs in visualization has proven to be a valuable tool, yet glyph design is a challenging task, with many aspects to be considered. Borgo et al. [27] discuss guidelines, techniques and possible applications that cover the foundations of glyph design. They provide information for the data mapping process, as well as categorizations of different glyph types. A similar work has been presented by Lie et al. [28], giving a concise overview of important design aspects. They introduce a two-dimensional glyph to visualize three-dimensional data and propose solutions to occlusion problems.Our Q-Glyphs are a special variant of the well-known star glyphs, which are especially suited to represent multivariate data. However, they lack the ability to map a central value when the axes are interpreted as directions, which is the case in our setting. Some of the more recent works include the one by Fuchs et al. [29], who focus on the contour influencing glyph similarity perception, and Miller et al. [30], who evaluate different ordering strategies for the data axes.

Ensemble Visualization: A recent survey of the concepts and approaches to ensemble visualization is given by Wang et al. [31]. Aggregation is named as a key, but optional, step in the pipeline. We did not deem aggregation over the ensembles necessary, since pre-filtering and restirction to certain scenarios already reduced the data to an acceptable amount. Similar to our approach of stacking glyphs, Tominski et al. [32] present stacked color-coded bands to visualize multivariate trajectory data sets. Ferstl et al. [33] similarly use the height dimension to draw stacked contour variability plots of weather data. They incorporate time information and create hierarchical grouping schemes including so-called space-time surfaces.

III. Visual Policy Analysis

The aim of our work is to facilitate neural network (NN) action policy *certification*. We want to enable engineers to get an overview and effectively identify potentially problematic regions (where policy behavior is unsafe) for closer inspection. The latter can go hand in hand with other tools like verification of local behavior [7]. Identification of problematic regions provides input for re-training. If intensive analysis does not exhibit weaknesses, in particular in regions engineers know to be potentially critical (like tight situations for collision avoidance), then confidence has been built for certification. We believe that visualization should be the core of a tool framework supporting engineers in such a certification process.

Previous work on action policy visualization focused on understanding the learning process itself, or explaining the decisions of a NN in a complex game like Go. A key difference to these is the global nature of certification: the user should, ultimately, inspect policy behavior across the entire state space. This is of course impossible for extremely complex games like Go (10^{171} states). But many actual applications are not that complex in terms of combinatorics, and even if their overall combinatorics is prohibitively high, they may still contain important dimensions along which they can be organized, and explored by visual overviews serving to identify local state-space regions for deeper (visual or otherwise) analysis. A prime example is physical location and velocity in cyber-physical systems. This dimension is key to safety, and is amenable to human inspection. Consequently, the present work focuses on obstacle avoidance in discrete 2D navigation.

While previous work focused on the visualization of individual traces, our visual analysis concept is state space driven. The visual entry point is an overview of the state space. To reduce the vast amount of information we exploit selection and summarization approaches. The engineer can select state space regions known to be challenging, e.g. due to narrow passages or sharp corners. To show different levels of information detail embedded in the selected region, we designed three glyphs (cf. left of Figure 1). The simplest glyphs just show the chosen action which is the common approach in previous works but does not give further insights into the predicted Q-values. Therefore, we designed an aggregation glyph summarizing the Q-values over all actions as well as a glyph variant showing all action Q-values in a compact visual representation – we call them Q-Glyphs. For states (position/velocity pairs) where the predicted Q-values are unintuitive, the user can use the tool to generate a trace (execute the policy) starting at the state of interest as shown in the middle of Figure 1. A detailed visualization then provides information about this trace, including the chosen actions and Q-values of the underlying NN.

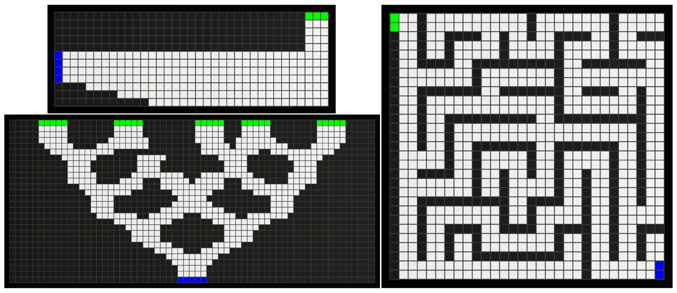
Fig. 2. Three established Racetrack maps: *Barto-small* (top-left), *River* (bottom-left) and *Maze* (right). Black regions are walls/boundaries; white cells are free positions. Start and goal cells are colored blue and green, respectively.

Due to the large amount of parameters controlling the Q-learning training process, a multitude of policies can be generated for a single use case. Focusing on only one such policy will leave potential insight undiscovered, so we have to provide an efficient way of comparing multiple policies. We achieve this by simultaneously showing multiple glyphs per state, each representing a different policy. While policies can be compared by arbitrary means, we propose some well-defined scenarios an engineer might find useful: *Train*, *Cross* and *Hyper*. The *Train* scenario is an often pursued goal of visualizing NNs, by showing the policy at different training stages and, as such, reflecting the learning progress. In *Cross*, we compare policies with varying initial parameters like the seed, as it has been shown that reinforcement-learned action policies are subject to brittle behavior [34], [35]. For a specific use case, one can also find settings for the agent or world that affect the final learning state, like noise probability. In the same sense we can also vary the training hyper parameters to produce *Hyper* scenarios. For Q-learning, the discount factor γ comes to mind, as it has a large impact on the quality of the resulting policy and is not trivial to choose correctly.

We situate our work in the Racetrack benchmark as a concrete case study. Racetrack is a common benchmark for MDP algorithms in the AI community [36]–[40]. It consists of a two-dimensional grid, where each grid cell is either a start or goal position, a free road position, or a wall. The car starts with a given velocity, usually 0. The objective is to reach the goal without crashing into a wall. The agent's actions are to accelerate by one unit in any of eight discrete directions, or to maintain the same velocity which we will refer to as the *zero-*

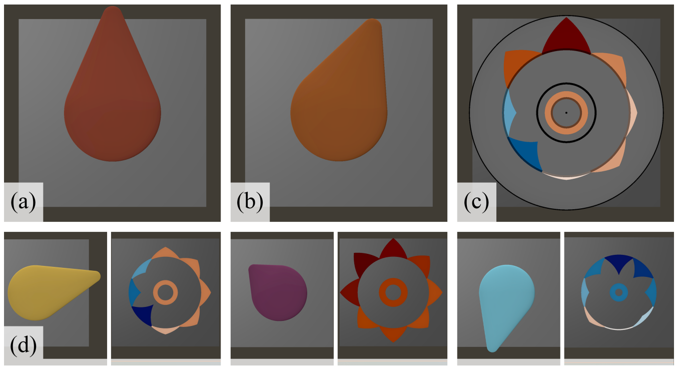

Fig. 3. The three different types of glyphs used in our visualization showing the same state: (a) *Chosen Action*, (b) *Directional Aggregation* and (c) *Q-Glyph*. More examples for *Directional Aggregation* are shown in (d), with associated Q-values given by *Q-Glyphs*.

action. The road is slippery, meaning that the action may fail to achieve its effect, so that the car keeps going at unchanged speed (compounding the risk to crash into a wall).

As Racetrack is defined over 2D space, we could restrict our visualizations to a 2D view for simplicity. Yet, we opt to use of the additional dimension as space for enabling policy comparison. Glyphs are stacked on top of each other, where each individual slice corresponds to a separate policy.

In our studies we use the track shape introduced by Barto et al. [36] as shown in Figure 2 (top-left), as well as the *River* (bottom-left) and *Maze* (right) maps introduced by [41]. Each state is defined by the 2D position of a cell and a 2D velocity vector over integer numbers[1].

IV. Glyph Design

To facilitate fast and comprehensive policy inspection, we designed three glyphs with increasing level of visualized information: *Chosen Action*, *Directional Aggregation* and *Q-Glyphs* (cf. Figure 3). To quickly gain an overview, viewing the *Chosen Action* might be better suited than directly showing all the individual Q-values. It can be used to identify regions of bad decisions, i.e. actions that directly result in a crash. Still, not all actions are chosen as clearly as may seem at first glance, as the Q-values of the other actions could be equally high or low. The *Directional Aggregation* serves the purpose to incorporate this information into an otherwise still simple glyph shape, to serve as an extended overview visualization. Finally, for a full detail view, the *Q-Glyph* maps all Q-values to a concise and compact representation. Our glyph designs are based on the following goals:

- *compactness:* Use space efficiently, while providing sufficient area to convey color information,
- *situatedness:* portray directional features in the glyph shape to facilitate implicit perception of direction, and
- *stackability:* keep design flat to allow stacking of glyphs.

[1]The state space is infinite in principle as velocities can get arbitrarily high, but we apply a velocity limit which obviously makes sense (speed limit).

A. Data Mapping

Our rewards for reinforcement are defined in the range $[-100, 100]$. Even though Q-values necessarily have the same maximal range, the Q-values predicted by a neural network can differ significantly. In the policies we analyzed, positive Q-values went only slightly above 100, but negative Q-values went down to values below -1500. Together with the domain experts we decided to clamp the Q-values to the reward range $[-100, 100]$ before mapping to glyph parameters, yet we leave the option to use the actual range of Q-values predicted by the policy for mapping.

To simplify color scale and glyph design we follow the typical windowing approach described by Lie et al. [28], by mapping Q-values to the range $[-1, 1]$. For this we introduce *windowed Q-values* ω for each mapping mode computed from the per action Q-values $q_{i=1..9}$ as follows:

$$\omega_i = \begin{cases} clamp\left(\frac{q_i}{100}, -1, 1\right) & \text{clamped range} \\ \frac{q_i}{q_{max}} & \text{full range} \end{cases}, \qquad (1)$$

where q_{max} is the absolute value maximum of all Q-values in the inspected policy or policy ensemble.

B. Chosen Action

For a fast inspection of policy behavior over a large part or the complete state space, visualization of the action chosen from the largest Q-value is sufficient. Arrows are the canonical choice for this purpose, as actions in the Racetrack context directly correspond to acceleration directions Color coding is natural to make directions distinguishable. However, while this would suffice to convey the directional information, choosing the right ratio between length and thickness poses problems, as space inside a single grid cell is limited. Thin arrows provide a better directional impression but deliver less area for color to be visible and vice versa. This is especially important as the individual glyphs become very small for zoomed out views so the visualization has to rely heavily on color. Furthermore, the zero-action cannot be visualized as an arrow.

Considering this, we opted for a raindrop-shaped glyph – represented by a flattened cone with round caps – with a large base and small tip (cf. Figure 3 (a)), where the tip position indicates the acceleration direction. For the zero-action the glyph tip is moved orthogonal to the track plane yielding a circularly symmetric shape.

Color is used as an additional visual cue to encode direction. We map 2D direction via angle to a perceptually uniform cyclic color map from [42] as shown in Figure 4 (b) and the zero action to black. This specific color map was chosen because it provides a large number of individually discernible colors and does not suffer from the problems induced by, e.g. a rainbow color scale. Mapping to color helps to identify regions with similar action decisions and visually separates different actions, which is especially useful in zoomed out views, where a large amount of glyphs is visualized (cf. Figure 8 and 10).

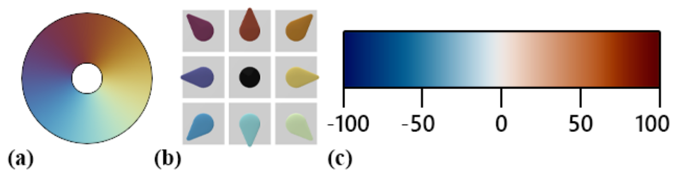

Fig. 4. The two color scales used for our visualizations: (a) *romaO* as used in the *Chosen Action* and *Directional Aggregation*, (b) all nine *Chosen Action* glyph variants arranged according to actions and (c) the diverging color scale *vik* used for *Q-Glyphs*. Color scales from [42].

C. Directional Aggregation

Showing the chosen action alone essentially ignores all other Q-values for a given state, hiding information about the tendency of the policy in that state. To provide a visual impression of the Q-value distribution over the actions, we devise an aggregation scheme to condense the information of all actions into a single directional representation, allowing to see at a glance how "certain" the decision of a policy is. Interpreting the Q-values of the NN output layer as the confidence to execute a specific action, we use them to weight the associated accelerations aggregate them.

Let $\vec{a}_i$ be the accelerations assigned to each action with $a_i \in \{-1, 0, 1\}^2$. Using the windowed Q-values as calculated in Equation 1, the final aggregated direction is determined as

$$\vec{d} = \frac{1}{9} \sum_{i=0}^{9} \omega_i \vec{a}_i. \tag{2}$$

Following this, the actions with negative Q-values will be aggregated with an inverse direction, emphasizing the tendency of the policy to not accelerate in this direction.

Visualizing the aggregation result is done by using the same primitives as in Subsection IV-B. The aggregated direction is mapped to its main axis with its length $\|\vec{d}\|$ being mapped to the distance between the base and the tip. We need to ensure a minimal distance to prevent the drop shape collapsing to a circle for small distances (this is only allowed for $\|\vec{d}\| = 0$). Therefore, with r_0 and r_1 being the drop's radii, we define the final length as $\|\vec{d}\| + |r_0 - r_1|$. Direction angle is mapped to color using the same color map as the chosen action. If the aggregated action has a length of 0, the glyph is again pointed orthogonal to the track plane and colored black. Examples of the glyph are shown in Figure 3 (b) and (d).

D. Q-Glyphs

So far, information about the Q-values has been reduced or accumulated before mapping it to simple glyph shapes. We can go one step further and show all the Q-values attached to a certain state, allowing for a more in-depth analysis of the confidence of a policy in its decisions and therefore enabling to better judge its safety. A more intricate glyph design is required to convey all of this information. For this, we take inspiration from star glyphs [29], [30], which are well-suited to encode higher-dimensional data in a compact representation, and use multiple data axes oriented around a central point, to portray each action Q-value. In the Racetrack context, a total of nine actions have to be considered, each of which stands for an acceleration in a certain direction. Hence we use the placement of the data axes as a visual metaphor for the directions. When placed on the grid, these axes will then correspond to the acceleration directions. However, only eight of the actions have an acceleration $\neq 0$, so the zero-action cannot be incorporated in the same manner and, following the metaphor, has to be mapped to the center instead. This visually separates it from the rest of the actions, although its importance as a decision is no less than the other actions. To address this shortcoming, we propose the Q-Glyph (see Figure 3 (c)) as a 2D visual representation for Q-values. Since the traditional variant of Racetrack lives in a 2D world, representing the accelerations with a 2D glyph feels natural. Additionally, since the glyphs have no inherent relation to 3D space, representing them in 2D allows for easier perception [28].

Since all actions describe equally distributed directions around a central grid position the overall shape of the glyph is circular. To create space for the zero- and non-zero actions in a way that conveys directional information, we separate the glyph into a central and rim part. The Q-value attached to the zero-action is portrayed in the center, using a ring as its visual representation. The directions of the non-zero-actions are drawn as triangles, from here on referred to as spikes, arranged radially in the rim part. They are spaced uniformly, their order being governed by the direction they portray. Depending on the sign of the corresponding Q-value, the tip of a spike either points towards the center for negative values or outwards for positive values. While the spikes belonging to negative Q-values, strictly speaking, not point in the direction of their acceleration, this serves to illustrate the policies strong suggestion to not execute this action and instead choose one of the opposing accelerations. Still, their position alone allows for a direct assignment to the direction.

The Q-Glyph is composed from these building blocks by mapping the windowed Q-values (see Subsection IV-A) to attributes of the geometric primitives. To account for the Q-values being both negative and positive, we need to adjust the mapping to the axes lengths of the final glyph geometry. We define two sets of radii: $R^c = \{r^c_{min}, r^c_{zero}, r^c_{max}\}$ for the center and $R^r = \{r^r_{min}, r^r_{zero}, r^r_{max}\}$ for the rim. As there is no space between the center and rim, it holds that $r^c_{max} = r^r_{min}$. Further, they need to satisfy $0 \leq r_{min} < r_{zero} < r_{max}$ and $r_{zero} - r_{min} = r_{max} - r_{zero}$.

When generating the geometry, the windowed Q-values of the non-zero actions $\omega_{2,\ldots,9}$ are mapped to R^r. For the zero-action we map to a ring originating from r^c_{zero} that grows outwards for positive and towards the center for negative values. This is achieved by creating a ring from r^c_{zero} to $r(\omega_1)$ if $\omega_1 >= 0$, and likewise from $r(\omega_1)$ to r^c_{zero} for $\omega < 0$, with $r(\omega_1)$ being a function to linearly interpolate ω_1 to R^c. The center radii are used to scale the central circle, while the rim radii are used to control the spike tips. The base sides of the triangular spikes are always located at r^r_{zero}. For our visualizations we chose r^c_{min} to be equal to zero. Mapping parameters for the Q-Glyph are shown in Figure 5 (a).

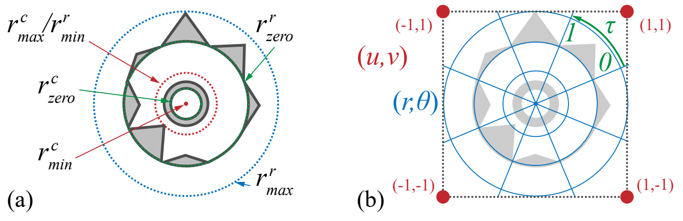

Fig. 5. Geometric parameters used for mapping data values to the Q-Glyph shape for (a) radii and (b) polar coordinate grid.

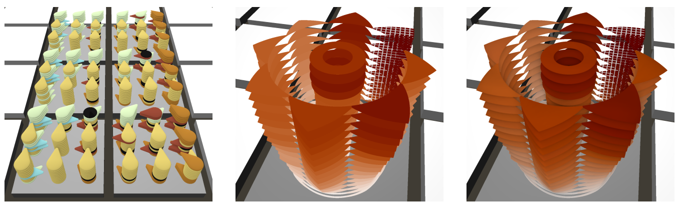

Fig. 6. Left shows an example of stacking *Chosen Action* glyphs. Middle and right show *Q-Glyph* stacks without and with ambient occlusion.

To avoid self-intersection of the circle r^r_{zero} with the triangular spikes, we define the spikes in polar coordinates.We ensure visibility of spikes with a Q-value of zero by giving the zero circle a finite thickness as can be seen in Figure 3 (d) in the right most example.

In addition to the shape, the Q-values are mapped to a diverging color scale as seen in Figure 4 (c), again accounting for negative and positive values, which helps to provide redundant information in the glyph and improves value perception [28]. We added a transparent grid pattern to the feature showing the highest Q-value, and therefore chosen action (see Figure 6 right), to aid identification for glyphs portraying similar values. We found this to be a good choice while keeping correct color impression. Readability of the Q-Glyph is increased by optional black guidelines that show the limits of R^c and R^r. Figure 3 (c) shows an example of a Q-Glyph.

E. Stacking

Stacking glyphs is straight forward. For each policy in the inspected ensemble, the glyphs are placed in their respective grid cell as before with the addition of being offset in the height dimension. Each slice of glyphs then corresponds to one policy in the ensemble. The policies are stacked from bottom to top based on a scenario specific parameter: for *Train* the number of performed training episodes, for *Hyper* the value of γ, and for *Cross* the seed value. In neither of these scenarios it is guaranteed that the top most slice shows the best (defined as having the highest goal probability over all states) policy in an ensemble. This even holds for *Train* scenarios, where policy quality can temporarily degrade with additional training runs. For this reason, we additionally support user defined or pre-computed scores to control stacking order.
Some perception issues may arise concerning the distinguishability of individual glyphs in a stack, especially for Q-Glyphs since they are only flat 2D shapes. Q-values often only differ by a small amount resulting in similar colors, which leads to two or more glyphs visually merging together (see Figure 6 right). To mitigate this, screen space ambient occlusion (SSAO) [43] is applied only to the glyphs. This darkens the spaces in between them and breaks up the uniform color look.

V. Visual Analysis Tool

This section discusses our prototype concept, giving details about the user interface and the two main analysis modes. The application itself is a plugin for the CGV framework [44] written in C++, which uses the OpenGL graphics API and provides means of ray casting based rendering of visualization primitives like boxes, spheres or tubes.

A. Data Acquisition

Reinforcement learning of our NN policies was done using PyTorch [45], with their C++ interface LibTorch being used to load exported action policies into our tool. The simulation environment and game logic was re-implemented to collect per state (position and velocity) the additionally necessary inputs of 8 directional distance values to the track boundary and three distance measurements to the nearest goal position. Feeding the input vector through the network yields the Q-values for all nine possible actions. Even though LibTorch is not thread-save in general, we could execute policy evaluation in parallel resulting in highly responsive evaluation of even large state space regions (cf. Subsection VI-A).

B. Q-Glyph Rendering

Q-Glyphs are generated directly on the GPU, with only 12 floating point numbers being transmitted per glyph (3D position and Q-values).The draw call is invoked using point primitives. A geometry shader generates a two-triangle quad covering the glyph, with uv-coordinates in the range $[-1,1]^2$ attached to the corners (see Figure 5 (b)).

Glyph geometry is generated in a pixel-perfect fashion in the fragment stage. The interpolated uv-coordinates are converted to polar coordinates (r,θ), with distance r and angle θ. Radius r is used to test if the fragment lies inside the center part, and subsequently the ring for the zero-action, or rim part of the glyph. For the spikes, a local interpolation parameter τ is calculated from θ to interpolate the radius as determined from the windowed Q-value ω_i according to a hat function – due to the polar coordinates this results in the curved appearance. In a last step, we test if r lies on one of the guideline rings and modify the color accordingly.

C. Track Visualization

As stated in Section III, a Racetrack map is defined as a two-dimensional grid with cells being defined as either start, goal, wall or free. The visual appearance of our track is inspired by Gros et al. [6]. Cells are rendered as boxes to visually separate them from one another and allow individual color mapping. They are color coded by type, with start/goal cells being blue/green, walls in dark grey and free cells in white. A black rectangular border displays the track boundary.

To enable the selection of velocities, a separate grid of boxes is drawn as an overlay in the top right corner (see Figure 1 left). The position of each box is used to encode its represented velocity, with $(0, 0)$ being located in the center. Rotating the overlay based on the current view angle with respect to the track helps to visually relate the selected velocities to their directions on the track and allows for easy interaction.

D. State Space Visualization

As discussed in Section I, the visualization of the state space of the system is an important factor to gain insight to the policy behavior. For this, we consider the state space as a set of distinct combinations of start positions and velocities. Displaying that set of configurations is accomplished by using the three glyph-based visualizations from Section IV, resulting in renderings like the ones shown in Figure 1 – called *Overview Mode* in our tool. The placement of each glyph is governed by the components of its attached state. If only one velocity is selected, the glyphs are positioned directly over the grid cell corresponding to the state position. If more than one velocity is chosen, the glyphs are positioned inside of the track cells to represent the layout as given in the velocity selector, their scaling being adjusted to fit all glyphs. In other words, their position inside a single grid cell relative to the cell center encodes the velocity they represent (cf. Figure 1 (right)).

E. Viewing Individual Traces

The different state space visualizations help to get an impression of the policy behavior for a large amount of start configurations. Once a region of special interest is identified, it can be helpful to get an even deeper look using the *Interactive Mode*. We support this by enabling the simulation of a full trace for a given start configuration, which is run until it reaches an end configuration. Traces are rendered using a combination of primitives – an example is shown in Figure 1 (middle). Spheres mark the position of individual states and are connected via tubes, showing the movement path of the agent. Additional information is shown for the currently selected state (marked with a yellow arrow), namely the velocity (blue) and acceleration vector (orange) that resulted from the applied action. Q-values are depicted above the states using *Q-Glyphs*.

F. User Interaction

Upon start, the user is presented a view of the map along with the velocity selector and a color legend. From here, interaction follows a top-down approach, starting with the state space visualization. To create a set of start configurations, positions and velocities have to be selected. For positions, the user can choose to select all valid (start and free) grid cells or only a subset through pointer interaction with the track. Velocities are chosen using the velocity selector following the same principles. The policy is then evaluated for the selected configurations directly within the tool following the procedure explained in Subsection V-A. As a result, we now have data for all nine Q-values for each selected configuration that can be displayed using one of the three presented visualizations from Section IV. When viewing ensembles, the user can select to only show a subset of the available policies.

At any time, the user can evoke the evaluation of a full trace for a single policy via direct interaction with a glyph, choosing the glyphs state as the start configuration of the trace. The visualization then changes to the mode described in Subsection V-E, allowing the user to follow the trace by iterating through the states and inspect the Q-values.

TABLE I
TIME IN MILLISECONDS FOR PARALLEL EVALUATION OF AN ENSEMBLE OF 10 NN ACTION POLICIES ON FREE TRACK CELLS FOR INCREASING AMOUNT OF VELOCITIES DENOTED BY n^2 WITH $n = 1, 3, 5, 7, 9, 11$.

data set	**free cells**	1	3^2	5^2	7^2	9^2	11^2
Barto-small	233	13	93	257	509	841	1254
River	486	24	192	531	1024	1701	2557
Maze	606	29	238	663	1287	2133	3228

VI. EVALUATION

This section provides the evaluation of our tool on the Racetrack example in form of performance measurement, a user study and a case study.

A. Performance Analysis

The application needs to achieve a certain level of efficiency to deliver a highly responsive and interactive user experience. This concerns both the time needed for evaluating the action policy as well as the render time. Tests were performed on a system with an Intel Core i9-9900K (8 $\times$ 3.6 GHz), 64 GB RAM and an Nvidia GeForce RTX 2080 Ti (11 GB of VRAM), with a Full HD view port fitting the whole data set. Evaluation was done on three tracks, *Barto-small*, *River* and *Maze* (cf. Figure 2), and ensembles consisting of 10 policies.

Table I shows consumed milliseconds for parallel evaluation of the policy on the three tracks used for evaluation. We utilized 16 threads running on 8 CPU cores. The evaluated state space regions were defined over all free grid cells with the number of velocities varying from 1 up to 11^2, where the latter includes 121 velocities in $\{-5, -4, ..., 4, 5\}^2$. Using the ensemble of 10 policies, this corresponds to 733k states on the *Maze* data set that can be evaluated in only 3.2 seconds. We consider this a worst case scenario, since it is unlikely for a user to request such large velocity distributions for all track positions. Given a more typical Racetrack benchmark like *Barto-small*, a state space of size 233×5^2 took around 256 ms to fully evaluate the ensemble. Typical scenarios will focus either on a reduced set of velocities and/or only a limited number of selected track positions. Execution time approximately grows linear in the number of evaluated states and parallelization achieves a speedup of a factor of 9 to 10.

Rendering performance proved not to be a limiting factor, with the largest test performed again on the 733k states of the *Maze* data set. Here we achieved average frame times of 5 ms for *Chosen Action* and *Directional Aggregation*, as well as 4 ms for the Q-Glyphs. Overall the results show high efficiency of the rendering process, delivering fully interactive

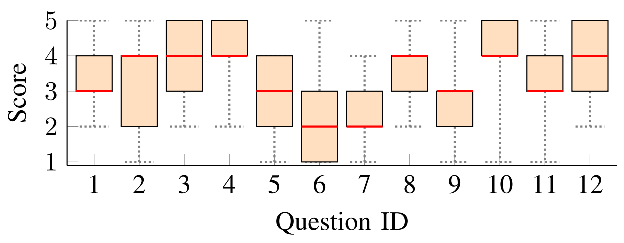

Fig. 7. User study results of the glyph and usability evaluation, red: median, gray: min/max, blue boxes: 0.25 and 0.75 quantile.

frame rates for all configurations. As even the frame rates in our largest test scenario are above 60 Hz, the tool can easily be used on moderately equipped systems.

B. User Study

We ran a user study to evaluate glyph functionality, how well users can achieve analysis tasks using our tool and general tool usability. The study was based on several different policies for the three test maps. We recruited 20 test persons, all with the necessary knowledge about MDPs, action policies, and NN training. They were briefed with a user manual containing tool and study instructions. Each user addressed four distinct tasks: (i) analyze individual policies based on our glyph visualization, determining regions where the policy is weak or dangerous (and hence needs to be re-trained); (ii) compare several policies based on our visualizations, thus gaining insights in the *Train*, *Cross* and *Hyper* scenarios; (iii) rate our glyph types with respect to their functionality, (iv) and rate the general usability of the tool. The full user instructions and questions asked are detailed in the supplementary material.

In part (i), each question was multiple-choice, i.e., users had to select the correct answers from a given set of options. Overall, correct answers were chosen with a rate of 84%. Users were mostly able to identify relevant phenomena like judging policy quality or finding regions where the policy is undecided.

In part (ii), the users were confronted with policy ensembles from the *Train*, *Hyper*, and *Cross* scenarios. In *Train*, users had to use *Q-Glyphs* to identify regions where policy quality drops after additional training. They were first given candidate regions, of which 80% were identified correctly; then they were asked to identify such regions themselves, which yielded 95% correct answers. In *Hyper*, the study participants were asked to find the best γ-value using the *Chosen Action* and *Q-Glyph* visualizations. Out of 9 possible values to choose from, 43% of the users chose $\gamma = 0.95$ and 38% chose $\gamma = 0.99$ respectively. Both are good options, and the justifications provided by the users made sense throughout, with 0.95 being best for finding shortest paths while 0.99 leads to policies that get stuck less. In *Cross*, users had to compare policies learned by separate training runs with different seeds. Their task was to analyze where the different runs lead to quality differences. All users accomplished this task well and where able to draw conclusions about the brittleness of policy training with respect to re-runs with different seeds.

Overall, the results from parts (i) and (ii) are thus quite positive, showing that the test persons manage to accomplish policy-quality analysis tasks well using our visualizations. In part (iii) and (iv), users were asked to provide direct feedback on our tool. Specifically, 7 questions in part (iii) asked users to rate our glyphs, and in (iv) 5 questions asked users to rate general usability. The answers were on a scale from 1 to 5 where higher numbers are better. Figure 7 shows the results. The underlying questions refer to: 1. Ease of understanding glyph placement within cells; 2. Helpfulness of *Directional Aggregation*; 3. Ease of understanding *Q-Glyphs*; 4. Appropriateness of *Q-Glyphs* to display Q-values; 5. Ease of seeing glyph directions in a stack; 6. Ease of perceiving *Q-Glyph* chosen actions in a stack; 7. Ease of inspecting *Chosen Action* glyphs in a stack; 8. Adequacy of tool complexity; 9. Ease of use; 10. Need for technical support; 11. Integration of functions; 12. Consistency of tool. Overall the answers are quite positive, exhibiting difficulties mostly in the inspection of glyph stacks, which constitute the most advanced part of the visualization and thus are a natural pain-point for users that spent only 1-2 hours with the tool. Most users (55%) preferred the *Chosen Action* glyphs, with 30% (primarily the more experienced) choosing *Q-Glyphs* as their preferred glyph.

C. Case Study

We next illustrate three examples in a case study, of how an expert – certification engineer – can use our tool.

Selecting the Best Policy: We first aim to select the best policy extracted during the training run. As discussed extensively by Gros et al. [7], [9], the training curve may not always indicate correctly which point during the training procedure produced the best policy. Hence, we extract the policy frequently and check the behavior using our tool. Figure 8 displays the stacking of two policies: one at the

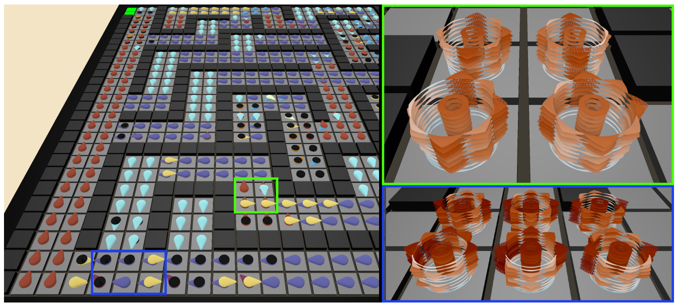
Fig. 8. Showing training progress on *Maze* using *Chosen Action* glyhs for two selected training stages. *Q-Glyphs* are used for detail views in selected regions for closer inspection.

end of training (upper slice) and a policy extracted after 70% of the training episodes (lower slice). Chosen actions are displayed for zero velocity. As one can easily find, there are several positions, especially towards the bottom of the map, where the agent already decided to steer the car into the goal direction, but decides to stand still after some additional training. Using *Q-Glyphs* to view full detail of the ensemble in largely undecided regions shows only little change in the

Q-values, yet learning tends towards the wrong action. Thus, using the tool, we conclude that the policy after the training is finished is not the best one, and the policy displayed in the lower slice should be used.

Selecting the Discount Factor: It is well known that deep reinforcement learning is brittle to its hyperparameters and that these must be tuned task-specifically. We make use of our tool to set the best value for the discount factor γ on the *Maze* map. For this, we view an ensemble of policies in accordance to the *Hyper* scenario. Additional filtering can be performed by stepping through the different discount factors. We find that for $\gamma = 0.65$, the agent firstly starts to learn at all, navigating to the goal line for all positions with $x \in \{1, 2\}$. For the rest of the map, all values of $\gamma < 0.95$ and $\gamma > 0.99$ seem to result in bad behavior in most regions. On first glance, $\gamma = 0.95$ and $\gamma = 0.99$ seem to be equally good options, such as already found by the participants of the user study. When e.g. considering the positions $(9, 15)$ and $(10, 15)$ for zero velocity, the chosen action (Figure 9 left) leads out of the dead end, which is a good decision. Still, considering the Q-Glyphs (Figure 9 right), we inspect that the agent might have learned avoiding the crash, but there is just barely a difference between the Q-values of the other accelerations. With these insights, we choose $\gamma = 0.99$ as the discount factor.

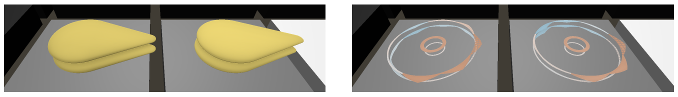
Fig. 9. Analyzing the discount factor γ on the *Maze* map.

Random Seed Influence: Deep reinforcement learning is not only brittle to its hyperparameters, but also to the used random seed. As the random seed is responsible for how the environment is explored in the beginning of training, the performance of the same training procedure can show big differences for different random seeds [46]. While this phenomenon can usually be measured by either considering the training curve or the agents quality only, our tool gives the chance to display the different behaviors with the help of *Cross* scenarios. Consider Figure 10, which shows the river map for zero velocity. The seed on the left first found the goal on the upper left. Thus, the agent is trained to navigate to said goal line, even though other goals are closer. This can be observed by considering the middle and right region, where the agent first navigates backwards to finally navigate to the goal in the upper left. The agent displayed on the right was trained with a different random seed. While in general this agent seems to navigate to the closest goal line, it has the drawback that there are some positions (black) where it decides to stand still.

VII. Conclusion and Future Work

Certification of neural network action policies is a quickly growing concern, bound to only become more important in the future. Visualization arguably has a key role in (almost) any technical solution, given the complexity of autonomous systems and their environments, and given the fact that full verification will hardly deliver safety guarantees for all possible situations. Therefore, to gain trust in an action policy, human engineers will have to understand its behavior and inspect its reactions against a large space of possible environment behaviors. Interactive visualization fits that purpose perfectly.

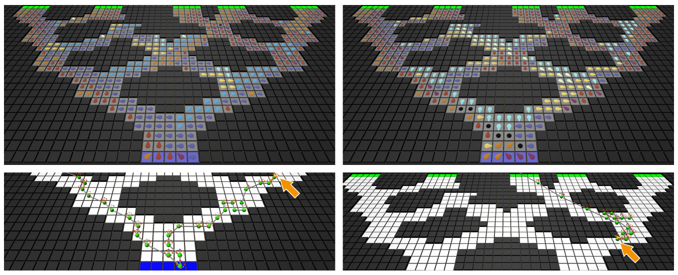
Fig. 10. Analyzing the seed influence on *River*. A trace started from the same state with zero velocity shows different behavior between the two seeds. The orange arrow shows the start position.

As concrete dimensions for organizing such a certification process, we believe that physical position and velocity are viable, even in systems that contain additional complexity (like internal state). Along with Gros et al.'s [6] work, our contribution forms a first step towards the exploration of visualization techniques in this context. Going beyond Gros et al., we introduced visualization concepts facilitating a deeper analysis of what has been learned. Our glyphs leverage a fast inspection of action policies on a state space level and enable detailed visualizations of the NN output layer, delivering a good value perception of output Q-values. As shown in the user and case study, *Q-Glyphs* help to better judge the Q-learning based action policies leading either to trust or the need to improve the training process. Finally, stacking allows to show whole policy ensembles, either illustrating the training process or comparing changes in hyperparameters. This greatly extends the amount of information that can be conveyed simultaneously, thus improving effectiveness of policy inspection.

It of course remains to carry these concepts to the much more complex systems and environments to which AI is and will be applied. To name just two examples, internal system state comes with a combinatorial complexity that needs to be visualized alongside physical position and speed; multi-agent behavior may be addressed with dimension reduction techniques and/or environment-behavior patterns, morphing back to 3D space in order to focus on individual agents.

Acknowledgment

This work has received funding from the Deutsche Forschungsgemeinschaft through DFG grant 389792660 as part of TRR 248, the Cluster of Excellence CeTI (EXC 2050/1, grant 390696704) and from the German Federal Ministry of Education and Research (BMBF, 01/S18026A-F) by funding the Center for Scalable Data Analytics and Artificial Intelligence ScaDS.AI Dresden/Leipzig. This work has been partially funded by the European Regional Development Fund (ERDF).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[2] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[3] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, 2019.

[5] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019. [Online]. Available: http://arxiv.org/abs/1912.06680

[6] T. P. Gros, D. Groß, S. Gumhold, J. Hoffmann, M. Klauck, and 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] T. P. Gros, H. Hermanns, J. Hoffmann, M. Klauck, and M. Steinmetz, "Deep statistical model checking," in *Proceedings of the 40th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE'20)*, 2020, available at https://doi.org/10.1007/978-3-030-50086-3_6.

[8] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[9] T. P. Gros, D. Höller, J. Hoffmann, and V. Wolf, "Tracking the race between deep reinforcement learning and imitation learning," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2020, pp. 11–17.

[10] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[11] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *CAV (1)*, ser. LNCS 10426, 2017, pp. 3–29.

[12] R. Ehlers, "Formal verification of piece-wise linear feed-forward neural networks," in *ATVA*, ser. LNCS 10482, 2017, pp. 269–286.

[13] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. T. Vechev, "AI2: safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy 2018*, 2018, pp. 3–18.

[14] J. Li, J. Liu, P. Yang, L. Chen, X. Huang, and L. Zhang, "Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification," in *SAS*, ser. LNCS 11822, 2019, pp. 296–319.

[15] F. Croce, M. Andriushchenko, and M. Hein, "Provable robustness of relu networks via maximization of linear regions," in *AISTATS*, ser. PMLR 89, 2019, pp. 2057–2066.

[16] C. E. Budde, P. R. D'Argenio, A. Hartmanns, and S. Sedwards, "A statistical model checker for nondeterminism and rare events," in *TACAS (2)*, ser. LNCS 10806, 2018, pp. 340–358.

[17] J. Bogdoll, A. Hartmanns, and H. Hermanns, "Simulation and statistical model checking for modestly nondeterministic models," in *Proceedings of the 16th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*. Kaiserslautern, Germany: Springer, 2012, p. 249–252. [Online]. Available: https://doi.org/10.1007/978-3-642-28540-0_20

[18] C. Baier *et al.*, "Lab conditions for research on explainable automated decisions," in *Proceedings of the 1st TAILOR Workshop - Foundations of Trustworthy AI - Integrating Learning, Optimization and Reasoning co-located with 24th European Conference on Artificial Intelligence, TAILOR 2020, Santiago de Compostela, Spain*, 2020.

[19] F. 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*, Jan. 2018.

[20] J. Wang, L. Gou, H.-W. Shen, and H. Yang, "DQNViz: A visual analytics approach to understand deep q-networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 288–298, Jan. 2019, conference Name: IEEE Transactions on Visualization and Computer Graphics.

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

[22] H. Joo and K. Kim, "Visualization of deep reinforcement learning using Grad-CAM: How AI plays Atari games?" in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–2.

[23] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 618–626.

[24] Y. Pang and T. Ito, "Visualization techniques to give insight into the operation of the Go policy network," in *2020 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2020, pp. 35–40.

[25] P. Hämäläinen, J. Toikka, A. Babadi, and C. K. Liu, "Visualizing movement control optimization landscapes," Sep. 2019.

[26] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," Dec. 2017.

[27] R. Borgo *et al.*, "Glyph-based visualization: Foundations, design guidelines, techniques and applications," in *Eurographics 2013 - State of the Art Reports*, M. Sbert and L. Szirmay-Kalos, Eds. The Eurographics Association, 2013.

[28] A. E. Lie, J. Kehrer, and H. Hauser, "Critical design and realization aspects of glyph-based 3D data visualization," in *Proceedings of the 25th Spring Conference on Computer Graphics*, ser. SCCG '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 19–26.

[29] J. Fuchs, P. Isenberg, A. Bezerianos, F. Fischer, and E. Bertini, "The influence of contour on similarity perception of star glyphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2251–2260, 2014.

[30] M. Miller, X. Zhang, J. Fuchs, and M. Blumenschein, "Evaluating ordering strategies of star glyph axes," in *2019 IEEE Visualization Conference (VIS)*, 2019, pp. 91–95.

[31] J. Wang, S. Hazarika, C. Li, and H.-W. Shen, "Visualization and visual analysis of ensemble data: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2853–2872, 2019.

[32] C. Tominski, H. Schumann, G. Andrienko, and N. Andrienko, "Stacking-Based Visualization of Trajectory Attribute Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2565–2574, Dec. 2012, conference Name: IEEE Transactions on Visualization and Computer Graphics.

[33] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann, "Time-hierarchical clustering and visualization of weather forecast ensembles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 831–840, 2017.

[34] R. Liessner, J. Schmitt, A. Dietermann, and B. Bäker, "Hyperparameter optimization for deep reinforcement learning in vehicle energy management." in *ICAART (2)*, 2019, pp. 134–144.

[35] B. Zhang *et al.*, "On the importance of hyperparameter optimization for model-based reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 4015–4023.

[36] A. G. Barto, S. J. Bradtke, and S. P. Singh, "Learning to act using real-time dynamic programming," *Artif. Intell.*, vol. 72, no. 1-2, pp. 81–138, 1995.

[37] B. Bonet and H. Geffner, "Labeled RTDP: improving the convergence of real-time dynamic programming," in *ICAPS*, 2003, pp. 12–21.

[38] H. B. McMahan and G. J. Gordon, "Fast exact planning in markov decision processes," in *ICAPS*, 2005, pp. 151–160.

[39] L. E. Pineda, Y. Lu, S. Zilberstein, and C. V. Goldman, "Fault-tolerant planning under uncertainty," in *IJCAI*, 2013, pp. 2350–2356.

[40] L. E. Pineda and S. Zilberstein, "Planning under uncertainty using reduced models: Revisiting determinization," in *ICAPS*, 2014.

[41] T. P. Gros, D. Höller, J. Hoffmann, M. Klauck, H. Meerkamp, and V. Wolf, "Dsmc evaluation stages: Fostering robust and safe behavior in deep reinforcement learning," in *International Conference on Quantitative Evaluation of Systems*. Springer, 2021, pp. 197–216.

[42] F. Crameri, "Scientific colour maps (7.0.1). Zenodo," Sep. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5501399

[43] M. Mittring, "Finding next gen – CryEngine 2," in *SIGGRAPH '07: ACM SIGGRAPH 2007 courses.*, 2007.

[44] S. Gumhold, "The computer graphics and visualization framework," https://github.com/sgumhold/cgv, accessed: 20-April-2021.

[45] A. Paszke *et al.*, "Automatic differentiation in PyTorch," 2017.

[46] T. P. Gros, "Tracking the race: Analyzing racetrack agents trained with imitation learning and deep reinforcement learning," Master's thesis, Saarland University, 5 2021.