AutoRL X: Automated Reinforcement Learning on the Web

LORAINE FRANKE, University of Massachusetts Boston, USA
DANIEL KARL I. WEIDELE, IBM Research, USA
NIMA DEHMAMY, IBM Research, USA
LIPENG NING, Harvard Medical School, USA and Brigham Womens' Hospital, USA
DANIEL HAEHN, University of Massachusetts Boston, USA

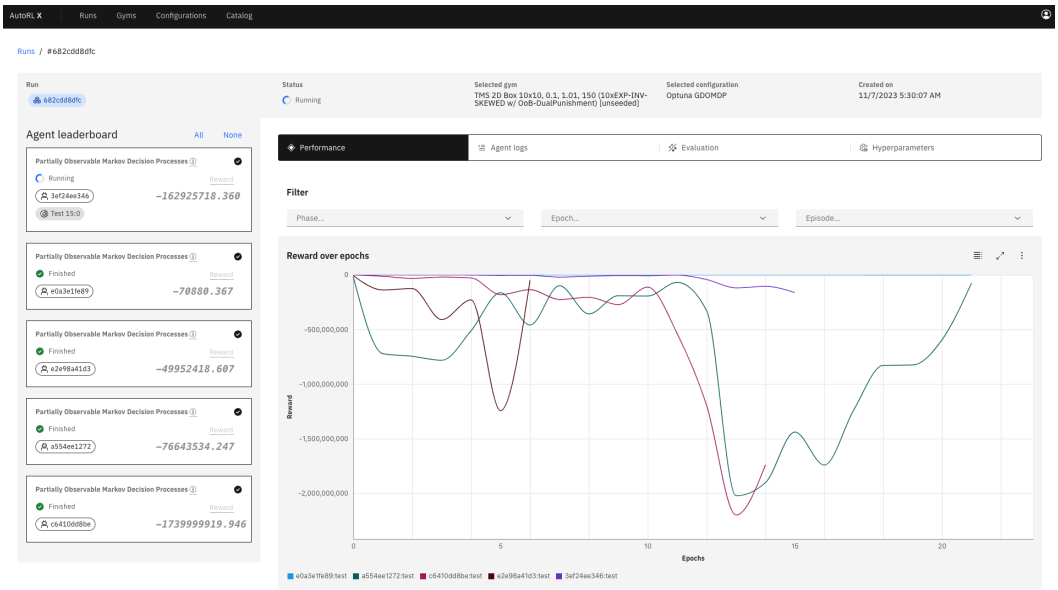


Fig. 1. Reinforcement learning agent evaluation in AutoRL X.

Reinforcement Learning (RL) is crucial in decision optimization, but its inherent complexity often presents challenges in interpretation and communication. Building upon AutoDOViz — an interface that pushed the boundaries of Automated RL for Decision Optimization — this paper unveils an open-source expansion with a web-based platform for RL. Our work introduces a taxonomy of RL visualizations and launches a dynamic web platform, leveraging backend flexibility for AutoRL frameworks like ARLO and Svelte.js for a smooth interactive user experience in the front end. Since AutoDOViz is not open-source, we present AutoRL X, a new interface designed to visualize RL processes. AutoRL X is shaped by the extensive user feedback

Authors' addresses: Loraine Franke, franke@mpsych.org, University of Massachusetts Boston, USA; Daniel Karl I. Weidele, daniel.karl@ibm.com, IBM Research, USA; Nima Dehmamy, nima.dehmamy@ibm.com, IBM Research, USA; Lipeng Ning, lning@bwh.hardvard.edu, Harvard Medical School, USA and Brigham Womens' Hospital, USA; Daniel Haehn, haehn@mpsych.org, University of Massachusetts Boston, USA.

2160-6455/2023/5-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

and expert interviews from AutoDOViz studies, and it brings forth an intelligent interface with real-time, intuitive visualization capabilities that enhance understanding, collaborative efforts, and personalization of RL agents. Addressing the gap in accurately representing complex real-world challenges within standard RL environments, we demonstrate our tool's application in healthcare, explicitly optimizing brain stimulation trajectories. A user study contrasts the performance of human users optimizing electric fields via a 2D interface with RL agents' behavior that we visually analyze in AutoRL X, assessing the practicality of automated RL. All our data and code is available online at: https://github.com/lorifranke/autorlx.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**; **Visualization**; • **Computer systems organization → Real-time systems**; • **Computing methodologies → Reinforcement learning**; **Artificial intelligence**.

Additional Key Words and Phrases: reinforcement learning, automation, decision optimization, visualization, RL agents, health

ACM Reference Format:
Loraine Franke, Daniel Karl I. Weidele, Nima Dehmamy, Lipeng Ning, and Daniel Haehn. 2023. AutoRL X: Automated Reinforcement Learning on the Web. *ACM Trans. Interact. Intell. Syst.* 1, 1 (May 2023), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Reinforcement learning (RL) stands as a cornerstone in the advancement of artificial intelligence, holding potential application in a multitude of domains through its ability to learn and optimize from interaction with arbitrary environments. The emerging field of RL is witnessing a rising demand for intuitive visual tools, specifically for understanding, validating and also increasing trust in machine learning. These tools are crucial in bridging the gap between RL algorithms and their practical application, enabling practitioners and those new to the field to understand, interpret, and interact with RL systems more effectively to solve complex problems.

In light of this need, we build on prior work in AutoDOViz [62], which exemplifies the advancements being made towards accessible and insightful RL visualization, empowering users to explore and dive into the complexities of RL processes by following the human-within-the-loop approach. Despite such advancements, the reality remains that many existing RL environments and simulations fall short of addressing the nuanced challenges posed by real-world applications. Furthermore, while more domain experts from various fields are incorporating methods of machine learning (ML), furthering access and applicability of RL seems to bear particular potential. The symbiotic relationship between human insight and machine efficiency—where machines can excel in optimization and computational tasks, human oversight and intervention ensure relevance and applicability to real-world scenarios.

In this work, firstly, we strive to democratize RL technology by leveraging open-source frameworks, thereby offering fully accessible code to the community. The spirit of open-source is embodied in the AutoRL X platform, ensuring a flexible architecture that can integrate with a variety of back-end engines. We aim to offer an RL experience on the web, making it straightforward and easy to use for a broader audience.

Second, we address shortcomings in user interface design identified in AutoDOViz paper. Thirdly we propose insights into a real-world use case in health care, for which we derive a reinforcement learning environment to investigate with AutoRL X. With detailed analysis, we wish to demonstrate real-world applicability of proposed platform. In this process, to study human accuracy and decision making on the proposed use case, we further conduct a comprehensive user study including domain experts in an interactive simulation component.

In essence, our work represents a significant leap forward from previous studies, harnessing the collective advancements in open-source technology and user interface design to bring RL closer

to solving tangible, complex real-world problems in various fields. The structure of the paper and workflow of this paper is presented in Figure 2, mapping out the journey and development of our system from theoretical framework to practical application in the dynamic landscape of reinforcement learning.

2 RELATED WORK

Reinforcement Learning and AutoML. Reinforcement learning (RL) is an influential method in machine learning for tackling sequential decision-making problems by training autonomous agents [28, 53]. These trained models, often termed agents, operate within specific environments and perform predefined actions. The concept of reinforcement learning offers a normative explanation deeply grounded in the psychological and neuroscientific viewpoint [50], elucidating how agents can enhance their mastery of an environment. Employing a trial-and-error methodology, RL aims to derive the best policy or behavioral approach [3]. The algorithm selects optimal actions based on feedback, typically in environmental rewards, to determine which actions benefit a particular state. In recent years, a variety of different algorithms and models have been proposed such as deep Q-networks (DQN) [39, 41], Deep DPG (DDPG) [35], Soft actor-critic (SAC) [19], Proximal policy optimization (PPO) [49] to name but a few. RL's effectiveness stems from its capability to navigate decision-making in unpredictable settings. This approach finds utility in diverse domains such as autonomous driving [60], robotics [36, 44], health, finance [47, 65], smart grids [68], gaming [40], space exploration [31], pedagogy [7] to name but a few [34]. Further, RL successfully handles challenges like multi-stage inventory management under demand fluctuations or autonomous manufacturing tasks that come with resource constraints. It minimizes not only redundant human engagements, such as system calibrations or monitoring but also ensures rapid system adaptability [48]. The primary objective of RL agents is to increase the overall expected reward by discerning the most suitable policy.

Recent advancements in data science and machine learning (ML) have streamlined critical workflows, such as data cleaning, feature selection, model training, and hyperparameter optimization (HPO) [29, 30, 32, 59, 69]. While many innovations address specific steps in this process, a fully automated and user-friendly approach remains rare. In this landscape, automated machine learning (AutoML) has emerged as a promising solution, extensively studied for its capability to enable machine learning with minimal human intervention [21]. Growing interest in fully automating the AI lifecycle has led to the concept of Automated Artificial Intelligence (AutoAI), a term often used interchangeably with AutoML. AutoML's objective is to automate the entire machine learning pipeline, from initial data pre-processing to the deployment of a fully trained and evaluated model. One of the most intricate tasks in AutoML, hyperparameter optimization, traditionally demands considerable expertise from data scientists and is highly dependent on the dataset. Consequently, automated HPO has become a focus within AutoML research, striving to minimize the necessity for deep AI knowledge or statistical skills, thus democratizing the creation of machine learning solutions [17]. This not only empowers specialists in various fields to engage with machine learning but also liberates data scientists from repetitive tasks, allowing them to dedicate more effort to enhancing visualization strategies—a critical aspect of the interpretability of machine learning models[61, 63]. The market response to these advancements includes a variety of commercial platforms like Amazon SageMaker [10], Azure AutoML [42], H2O AutoML [33], Google AutoML [5], and IBM AutoAI [51, 56, 63], each offering unique features tailored to different aspects of the machine learning process. Meanwhile, open-source tools such as AutoKeras [27], TPOT [45], Auto-Sklearn [14], and LALE [22] are expanding the accessibility and flexibility of AutoML for a broader audience. These automation principles extend into reinforcement learning (RL), where Automated Reinforcement Learning (AutoRL) seeks to facilitate the RL pipeline. RL's performance

is notably sensitive to the correct tuning of hyperparameters; thus, AutoRL focuses on adjusting these parameters automatically, which is essential for the progress of RL research and application. Despite its potential, AutoRL is not without its own set of hurdles and challenges, signaling an ongoing and dynamic field of study [8, 13, 15, 46, 52]. At the forefront of this initiative is ARLO [43], a pioneering framework for AutoRL presented in 2022. ARLO is a Python library for automating all the stages of an Automated RL pipeline. Unlike many other AutoML libraries, ARLO does not tie to specific RL algorithms, making it versatile and extensible. It further strengthens its foundation by relying on recognized open-source platforms and libraries like the OpenAIGym [6] and DeepMinds' MuJoCO [54]. Further libraries for automated RL include, for example, Alibaba's EasyReinforcementLearning [1] or IBM's AutoDO [37].

XAI and Visualization. Explainable AI (XAI) is essential in integrating advanced AI technologies such as computer vision and natural language processing (NLP) into the corporate sphere. A significant contribution to this field includes researching how visual tools can help demystify AI models and laying down fundamental terminologies and concepts. References to critical surveys in the literature [1, 23] underline the importance of visualization for comprehending and explaining AI algorithms. As we dive into Reinforcement Learning (RL), XAI reveals a tendency toward requiring visual aids and straightforward policy summaries to interpret complex algorithms and agents. Despite the progress, the challenges persist, especially the absence of user-centric studies that would anchor XAI techniques more firmly in the user experience. Additionally, the predominance of oversimplified examples in research, which fail to encapsulate the complexities of real-world applications, raises concerns about the applicability and robustness of these XAI methodologies. Wells and Bednarz have identified immersive visuals and symbolic representation as promising future research directions that could address some of these challenges in RL [64]. From the current literature, we can derive that one of the crucial trends for creating explainable and interpretable AI systems within the domain of Reinforcement Learning is visualization. This focus on employing visual tools not only aligns with the broader objectives of XAI but also represents a strategy for addressing the challenges specific to RL. In general, visualization is crucial for understanding complex machine learning (ML) algorithms, such as web-based visualizations, as emphasized by multiple studies [12, 16, 38, 58, 70]. Specifically, visual analytics for RL can enhance trustworthiness during RL training and evaluation processes. Current literature provides a variety of tools, such as ReLVis [48] to help in tracking RL experimentation, while DRLViz [26] and DRLIVE [58] offer insights into an agent's internal memory and interactive tracking, respectively. PolicyExplainer [38] allows direct queries to autonomous agents. Other tools, like DQNViz [57] and DynamicsExplorer [20], target specific RL algorithms or policies. Although direct visualization methods exist [18, 25, 41, 67], challenges remain, especially in multi-objective optimization scenarios [9]. The current landscape and literature, as mentioned above, indicates an evident gap in real-time, interactive RL visualizations. Therefore, this research bridges the gap between the need for real-time, interactive and also automated RL user interfaces and the current solutions. By integrating the strengths of past works and addressing the highlighted gaps, we aim to pioneer a robust and intuitive open-source platform for automated RL visualization and enriching the domain's landscape.

AutoDOviz. AutoDOViz [62] is an interactive platform designed to enhance the user experience in automated Decision Optimization (DO) with Reinforcement Learning (RL). Developed with insights from semi-structured expert interviews involving business consultants and DO practitioners from

[1] https://github.com/alibaba/EasyReinforcementLearning

a variety of industries, the system aimed to meet the design requirements essential for human-centered automation in the realm of DO. One of its main achievements is its ability to improve trust and confidence in ML, especially RL agent models. This was achieved by adding transparent presentation of reward metrics, making the complexities of automated training and evaluation processes both accessible and comprehensible to users. Furthermore, AutoDOViz incorporates the power of automated DO algorithms for RL pipeline searches, generating insightful policy data and visualizations. These advanced visualization capabilities facilitate more effective communication between DO experts and those from various domains. Another feature of AutoDOViz is its gym composer and the accompanying gym template catalog. This unique addition aimed to lower the entry barrier for data scientists when specifying problems for RL. With an array of pre-defined templates, users can easily find and reuse examples, speeding up the problem-definition process. However, the user study did reveal a hesitancy in contributing to this catalog, largely due to concerns about client confidentiality. The system also features a streaming architecture, introducing a novel Human-within-the-Loop approach for enhanced interactions. However, the study also identified areas for potential improvement, providing guidance for future iterations which we will pick up upon in Section 3.2.

3 SYSTEM DESIGN

Figure 2 outlines the development process of AutoRL X. Section 3.1 revisits the user interface of AutoDOViz, detailing insights that have been incorporated in AutoRL X. Subsequently, Section 3.2 discusses the requirements and key takeaways from the AutoDOViz experience, setting the stage for the subsequent section describing the architecture of our open-source platform.

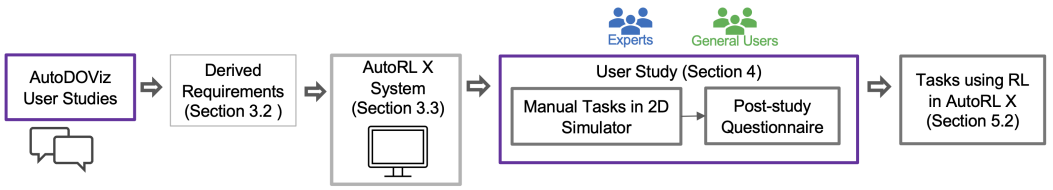


Fig. 2. Workflow: From AutoDOViz with its 3 different user studies we distill requirements described in Section 3.2 Next we design our prototype for AutoRL X. Lastly, we conduct a user study for a real-world use case that we then map to a reinforcement learning environment.

3.1 Insights from AutoDOViz

From exploratory interviews, user studies, and reflections from previous work in AutoDOViz [62], we were able to derive further features and suggestions that could drive our system design. Furthermore, since the AutoDO [37] engine is not easily accessible as an open-source software outside the IBM network for industry and clients, we propose *AutoRL X*, as an alternative option to visualize and interact with AutoRL on alternative engines. Different types of visualizations have been proposed for RL analytics, such as state space visualizations, action space visualizations, policy and value function visualization, training progress and convergence, or agent-environment interaction dynamics. Furthermore, we propose a set of interactive tools that can be tailored for RL visualization. For example, drag-and-drop features should allow users to input and then customize their own RL environment, or zooming features in the performance charts to focus on particular areas of the state space or time intervals, also visualizing real-time feedback for hyperparameter tuning and algorithm adjustments, and line charts to compare different RL algorithms (agents) and its

configurations side by side. In AutoDOViz, a dashboard provides users with a straightforward access to three core entities: environments (gyms), engine configurations, and executed jobs (Figure 3). This structured layout enables users, including business domain stakeholders, to trigger executions and access high-level visualizations of their RL experiments without delving into the technicalities of gym implementations. Moreover, AutoDOViz incorporates a configuration wizard that simplifies the complex process of setting up RL agents and their hyperparameters, and implements two further types of visualizations: transition matrices and trajectory networks, to present behavioral information about the agent to provide detailed insights and increased confidence to the user. The interface displays a list of selectable RL agents and, for each one, a detailed configuration panel that allows users to adjust hyperparameters, providing options for types, possible values for discrete parameters, ranges for continuous ones, and default values. Further, tutoring interface strategies of AutoDOViz are applied in the process of modeling the gym, where the composer led users through a series of decisions.

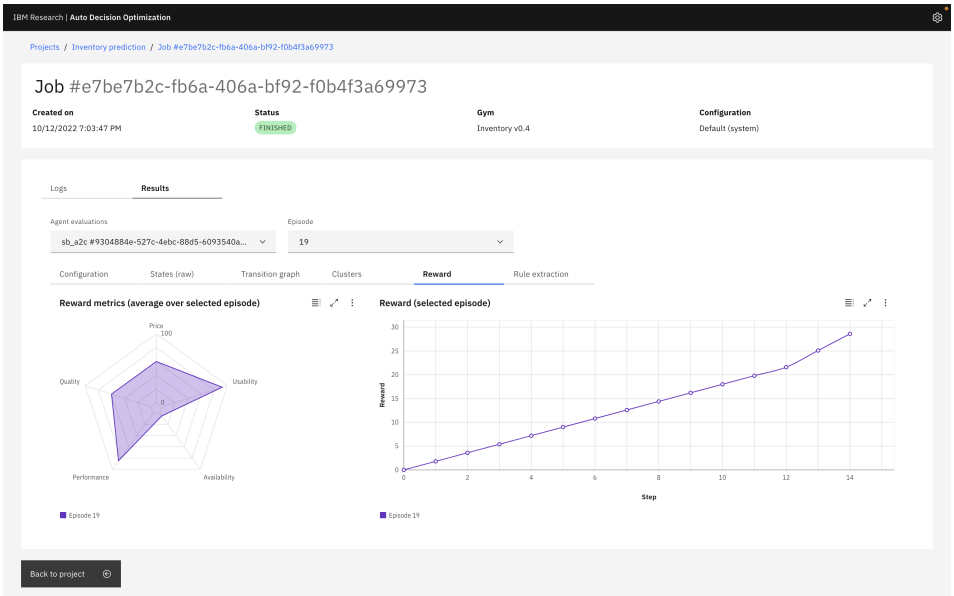


Fig. 3. AutoDOViz's Job View: The output is visually represented in a dashboard format.

3.2 Requirements

While our main priority in developing an open-source version was to maintain the functionalities that we offered users in the proprietary software, we still aimed to incorporate detailed findings and feedback from the three user studies we conducted for AutoDOViz. In the following section, we derive requirements that guide the development of AutoRL X, a more refined and user-centric follow-up system: First, the feedback indicated that the user experience for small screen-size users could be improved significantly to reduce scrolling in the composer screen. There is a need to ensure the system is optimized for various devices, including tablets and mobile phones (**R1**). The agent listing screen was identified as an area for improvement. Enhancing its layout, functionality, and filter options could provide a smoother experience (**R2**). One participant expressed a need for more understandable visualizations. Addressing this could involve using tooltips, legends, and

contextual guides to help users decipher visual data **(R3)**. Suggestions were made to incorporate time sliders to replay real-time feedback on agent progress visualizations. This feature would allow users to rewind, pause, and analyze agents' actions over time **(R4)**. For those less familiar with the tool, there is a need for additional on-screen explanations, tutorials, or a help section to guide them **(R5)**. Given that one user expressed interest in collaborating using AutoDOViz, introducing features that enable collaboration, such as shared views, commenting, and real-time edits, could be beneficial **(R6)**. Several participants showed interest in integrating AutoDOViz with their existing toolkits. Developing plugins or APIs to facilitate integration with popular data science and ML tools could enhance its adoption **(R7)**. Based on feedback, while users are keen on using pre-existing templates, there is a hesitance in contributing due to confidentiality concerns. A potential solution is to provide more generic templates or allow for anonymized sharing **(R8)**. Recognizing that preferences for working in shared vs. custom environments are highly use-case dependent, the system could offer more granular control over environment settings, with attention to security, privacy, and cost **(R9)**. While the UI was appreciated for its familiarity, maintaining consistency with popular data science software can ensure users find the platform intuitive **(R10)**. Since the UI successfully allowed data scientists to learn about DO tasks quickly, adding more educational tools, walkthroughs, or interactive demos might enhance user understanding **(R11)**. Emphasizing transparency, especially on metrics, was claimed essential by user study participants. Features that break down metric calculations or provide insights into algorithmic choices can boost user confidence **(R12)**. Table 1 lists all the requirements we could identify.

Table 1. Requirements for AutoRL X derived from AutoDOViz - A Human-Centered Approach

Requirement	Description
R1 Enhanced UX for Small Screens	Optimize platform for various devices to improve user experience, e.g. reducing scrolling on small screens like tablets and mobile phones.
R2 Refined Agent Listing	Improve layout, functionality, and filtering options of the agent listing screen for a smoother user experience.
R3 Improved Visual Interpretability	Use tooltips, legends, and contextual guides to make visual data easier to understand for users.
R4 Inclusion of Time Sliders	Integrate time sliders in visualizations to allow users to rewind, pause, and analyze agent actions over time.
R5 Enhanced On-screen Explanations	Provide additional on-screen explanations, tutorials, or help sections to assist users less familiar with the tool.
R6 Collaborative Features	Introduce collaborative features like shared views, commenting, and real-time edits to facilitate teamwork.
R7 Integration with Existing Toolkits	Develop plugins or APIs for integration with popular data science and ML tools to encourage adoption.
R8 Expand the Gym Template Catalog	Offer more generic templates and anonymized sharing options to address confidentiality concerns.
R9 Customizable Environment Preferences	Provide granular control over environment settings with a focus on security, privacy, and cost.
R10 Enhanced UI Consistency	Maintain consistency with popular data science software to ensure intuitive use.
R11 Educational Features	Add more educational tools, walkthroughs, or interactive demos to enhance user understanding of DO tasks.
R12 Transparency Enhancements	Implement features that provide insights into metric calculations and algorithmic choices to increase user trust.

3.3 AutoRL X Architecture

The architectural schema of the AutoRL X system is depicted in Figure 4. Informed by the identified requirements, the design closely mirrors that of our proprietary AutoDOViz platform. Similarly,

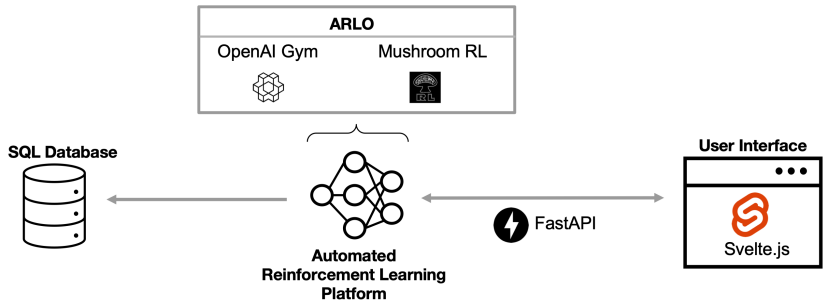


Fig. 4. **Architecture of AutoRL X:** The heart of the platform is an AutoRL engine to run different reinforcement learning pipelines. We opted for ARLO as reference implementation, which builds on OpenAI Gym standard with Mujoco 3D environment extension, while leveraging Mushroom RL agent implementations. Data, such as run and model configurations and logs from our logger, is stored in our SQL Database. FastAPI for REST offers flexible communication with any AutoRL framework, connecting to our UI developed with Svelte.js.

the system should manage three entities by the user of AutoRL X: gyms (or environments), engine configurations and agents, and resulting runs (or jobs). The system architecture is structured into the following three principal components:

Backend. The backend of our application uses an open-source AutoRL engine, ARLO [43], to facilitate automatic computation of reinforcement learning (RL) pipelines. ARLO handles OpenAI Gyms [6], MuJoCo [54] 3D environments, and leverages agent implementations from Mushroom RL [11]. It is suitable for diverse research and development scenarios. Figure 5 shows eight ARLO models offered through our UI. While we are focusing on Online RL scenarios throughout this work, next to DQN, PPO, SAC, DDPG, GPOMDP, the ALRO framework also features FQI, DoubleFQI, and LSPI for Offline scenarios. ARLO further provices different tuner strategies that our users can choose from in our interface. For example, a Genetic Tuner, which evolves a population of model configurations by mutation and selection to optimize hyperparameter for performance on a given evaluation metric. We also provide access to the Optuna Tuner [2], performing hyperparameter optimization by searching through a predefined space and evaluating model performance. It uses advanced algorithms to determine the best set of parameters with features like trial pruning and parallel execution to speed up the search. From a dropdown menu in the UI, the user can also choose different evaluation metrics for the RL pipeline, such as a discounted reward, which evaluates the average of cumulative rewards received over episodes, adjusted by a discount factor (*gamma*) to account for the time value of rewards. In contrast, another metric, temporal difference error, calculates the average squared deviation between predicted and actual rewards in subsequent states, reflecting the accuracy of the value function. Lastly, a time series rolling average discounted reward evaluates the average of cumulative rewards received over episodes adjusted by a discount factor to account for the time value of rewards.

Further enhancing the backend, our system is designed with highest extensibility in mind (**R7**). It is not limited to the proposed ARLO framework; the architecture allows for integration of alternative automated reinforcement learning engines. This is achieved by AutoRL X's robust logging mechanism that records run metadata and streams model logs into a database, ensuring a structured and retrievable data management process. Additional AutoRL frameworks can simply

[2]https://optuna.readthedocs.io/en/stable/index.html

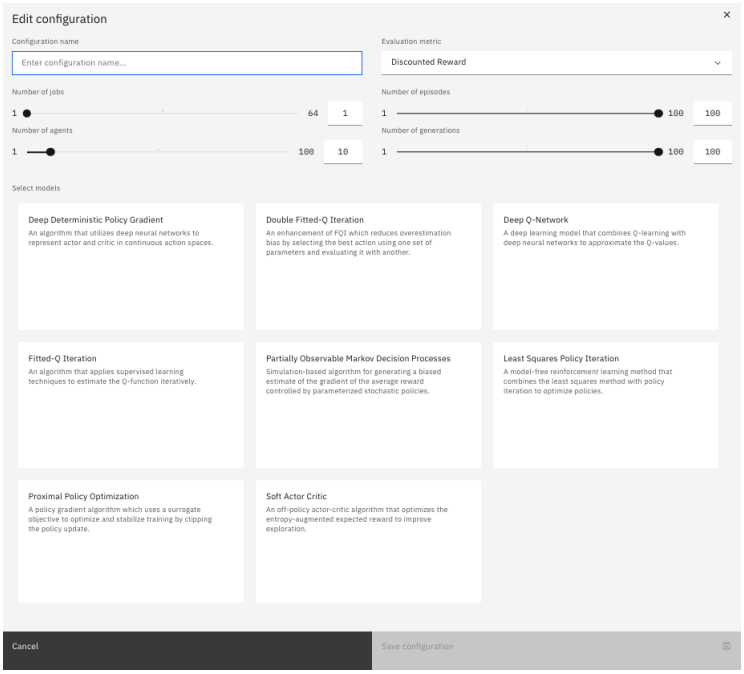


Fig. 5. **Model Selection:** Users can select from a variety of offline and online models and configure number of agents, episodes, number of generations and also select one of 3 offered metrics. Then the user can save them as configurations an then run these on RL environments.

be integrated by providing a job execution script (Python), and making them communicate with the REST API provided in AutoRL X.

API. The REST API built with FastAPI [3] offers execution of RL gyms in ARLO and integrates with the SQL database via an internal *service layer*. One part of the API is responsible for managing the database operations related to gyms, configurations, runs, and models. The services handle dataabse connections securely and perform queries and updates efficiently to ensure lazy loading on the front-end. For example, to get information about model run episode trajectories, we offer a `get_trajectory` method, which retrieves only the currently selected step sequence requested by the user in the UI. Our API further comprises of other parts like a logger with *zlib* compression support for fast write- and read operations, to enable an overall comprehensive and scalable backend solution.

Frontend. Our frontend framework of choice is *Svelte.js* [4], a modern and friendly framework [4] with *actual*, compiler-ensured state-reactivity from the bottom. Svelte stands out for its innovative approach to building user interfaces, leading to more efficient updates and cleaner code. This advantage lies in Svelte's departure from the virtual DOM paradigm, offering direct manipulation of the DOM and, thus, faster performance and a more straightforward development experience. To obtain responsive and user-friendly interface components, we add Carbon Design Framework. The *Carbon Components Svelte* library implements the Carbon Design System [5], an open-source

[3]https://fastapi.tiangolo.com/
[4]https://svelte.dev/
[5]https://carbon-components-svelte.onrender.com/

design system developed by IBM that emphasizes reuse, consistency, and extensibility. This design is tailored for complex, enterprise-level interfaces to accelerate the development process while maintaining a high standard of design quality and user experience. By utilizing this library, we were able to maintain a consistent look and feel across our application. Our decision was informed by insights gathered from interviews with data scientists using AutoDOViz, and many were already acquainted with the Carbon UI library from other software tools they use in their daily work. The familiarity of the UI framework contributed positively to the user experience. Participants noted that this consistency aided them in efficiently performing their tasks, fulfilling the user interface consistency requirement **R10** highlighted as a priority for our system's design.

In result, as shown in Figure 6, users of AutoRL X can now select from refined agents in a pipeline run according to **R2** with novel filter options to search for specific phases like learn or test phase, a certain epoch, filter through iterations and actions to closer examine agents behaviors. In line with **R3** and **R4**, we have added tooltips and time sliders to better retrieve information from the line charts allowing users to easier understand visual data. The user can also see which agents are still running or already finished. Next to the filtering options, we also added the possibility to more refinedly view agent logs, visited states, and hyperparameters that were demanded in **R12**.

In response to user feedback received in AutoDOViz, where users expressed confusion regarding navigation of categories in the gym catalog, we make a slightly improved design prototype in AutoRL X as seen in Figure 9. Specifically, in AutoDOViz, we categorized gyms based on the North American Industry Classification System (NAICS) into various business problem categories. We introduce a more user-friendly navigation system in AutoRL X, which guides users through the different gym categories via breadcrumbs to streamline the user experience and facilitate easier exploration. For requirement **R1**, we personally tested our platform on multiple devices: a tablet, smart phone and even the mixed reality browser offered in Meta Quest 3 by connecting via the local network. Overall, we were able to address 8 of the 12 requirements from our full list in this work. The remaining requirements **R5**, **R6**, **R9**, **R11** are mapped to Github issues in our open-source repository.

4 APPLICATIONS OF REINFORCEMENT LEARNING

RL environments range from simple board games to complex simulated robotics or virtual ecosystems, providing controlled spaces where agents can safely explore and learn from interactions. However, transitioning from these well-defined environments to real-world applications is often challenging: real-world scenarios are often unstructured and unpredictable, with noisy data and non-stationary dynamics that can significantly hinder performance of RL algorithms trained in more controlled settings. Moreover, ethical and safety considerations involved in trial-and-error learning methods raise substantial barriers to deploying RL agents in situations where mistakes can have serious consequences, such as in healthcare or autonomous driving.

In AutoDOViz, we have identified a variety of domains that are using optimization strategies (or decision optimization) by conducting semi-structured interviews with specialists in agriculture, automotive, government, manufacturing, oil, or retail industries. All these examples mainly focused on business applications, and the environments in AutoDOViz's gym catalog were inspired by these use cases. In this work, we aim to explore a real-world problem from the medical/healthcare field that could be solved with RL. We believe that specialists from this domain can further benefit from RL's capabilities. Reinforcement learning has shown promise in a range of medical applications, although it is still in its early stages of adoption in the field [66]. Some examples of RL medical and healthcare application successes include its use for critical care or chronic diseases like cancer, diabetes, and HIV or in automated medical diagnosis, especially for medical imaging [24]. Furthermore, RL can be used for surgical robots [55]. Other application domains for RL in healthcare involve clinical

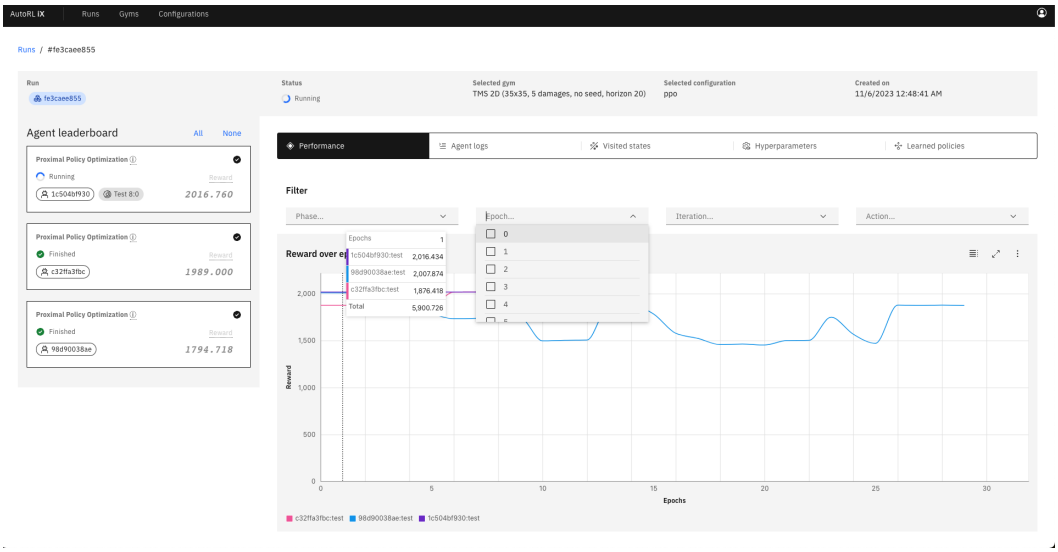


Fig. 6. **Agent Leaderboard:** Running an environment with PPO algorithm. User can select different agent configurations on the left and add them to the line chart to explore their progress. In the menu above the chart the user can filter by different phases, epochs, iterations and actions, beside the option to select different tabs with agent logs, visited states, hyperparameters and learned policies.

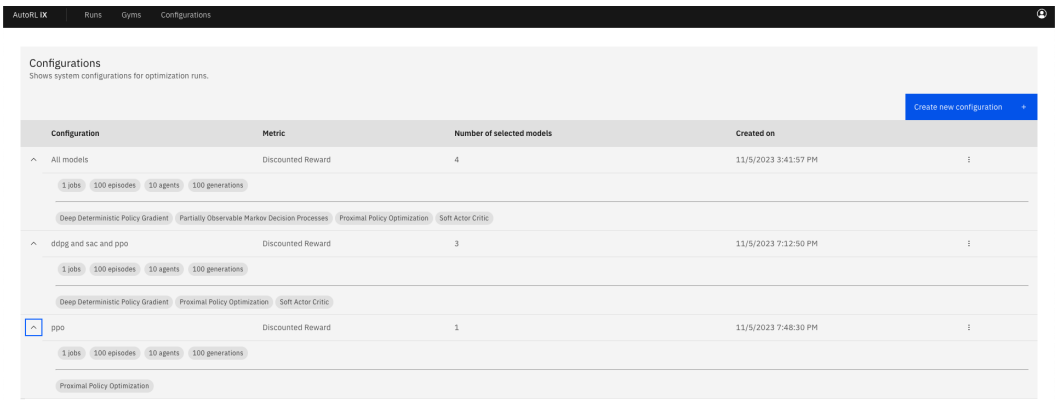


Fig. 7. **Configuration Management:** The Panel displays the interface for managing a collection of agent-model configurations, which are persistently stored within the system's database. These configurations can be subsequently retrieved for further analysis or modification.

resource allocation, controlling, or healthcare management. We opted for a problem that is visually appealing in order to be sufficiently comprehensible to the general reader, despite being very domain specific. This allows us to exemplarily journey together from problem inception to RL agent run results and inspection within AutoRL X. This health-related reinforcement learning challenge was further identified in concert with collaborators from the domain.

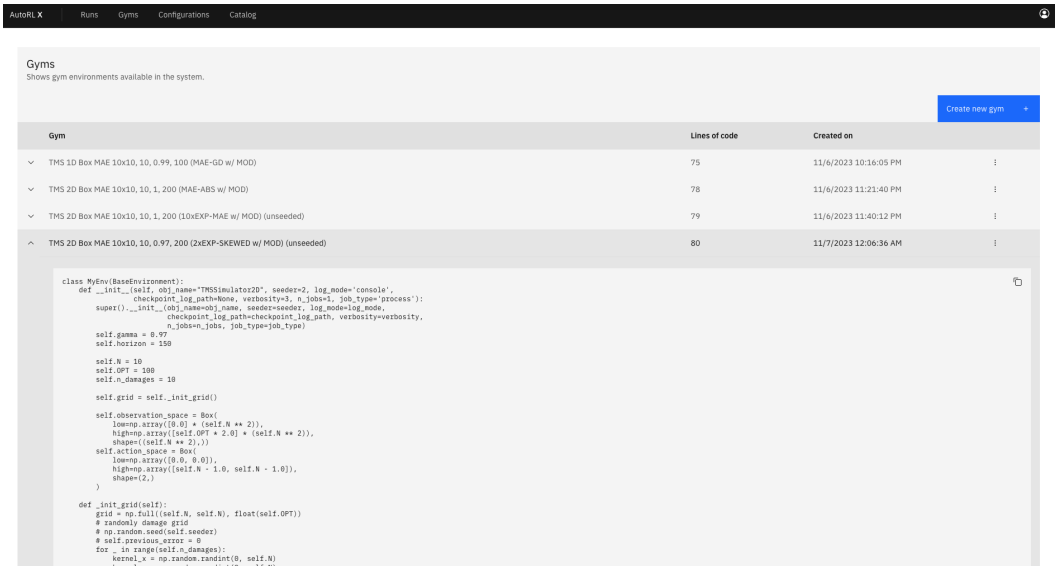


Fig. 8. **Environment Management:** The panel illustrates the environment registration and listings module, where users can add, name, duplicate and inspect the existing RL environments. These definitions are also preserved in the database, enabling continued access across sessions, including after the browser has been closed.

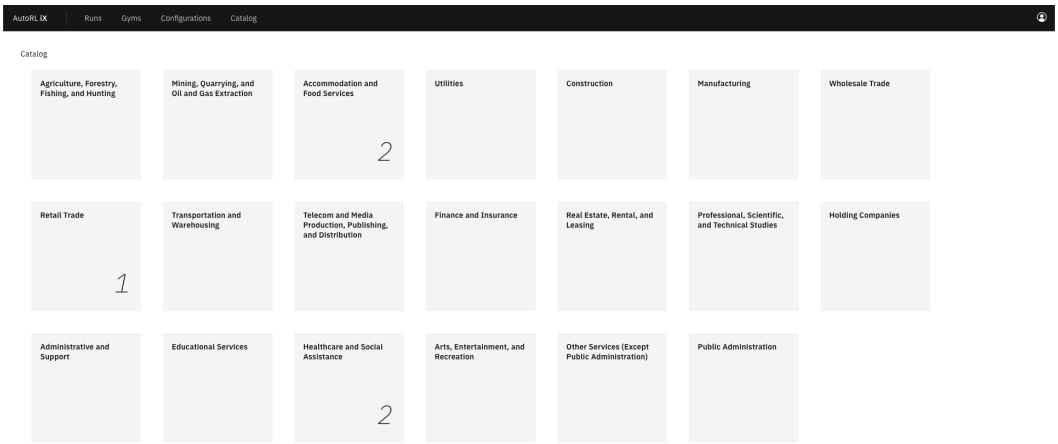


Fig. 9. **Gym Catalog:** The Gym Template Catalog of AutoRL X: Gym templates can be explored with breadcrumbs sorted by NAICS categories.

4.1 Use Case Scenario: Transcranial Magnetic Stimulation

We identify a real-world optimization problem that seems worthwhile to explore with reinforcement learning pipelines: Brain stimulation refers to the application of electrical, magnetic, or other forms of energy directly or indirectly to the brain to alter its activity, with therapeutic purposes such as treating psychiatric conditions or neurological disorders, or even to enhance cognitive function. There are invasive forms of brain stimulation where electrodes are placed onto the brain with

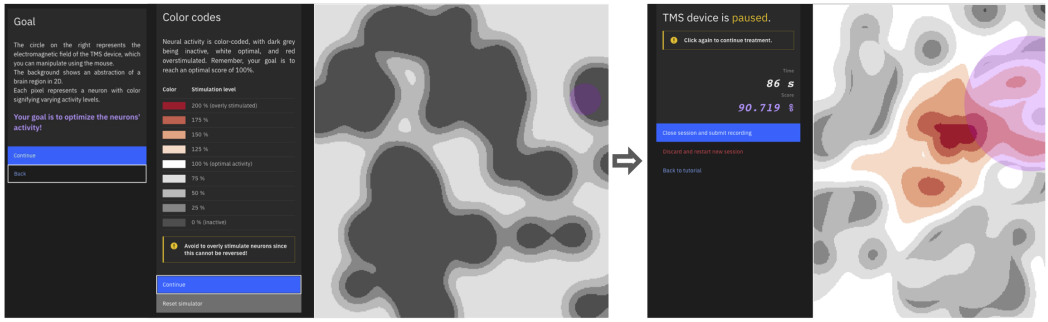


Fig. 10. **Interface of the 2D Simulator:** Users can click through the study. Left: Explanation of the goal is shown at first, then the color legend in the next page. Right: Users can see if the electric field (purple) is active or not, as well as the time and current score.

surgery or non-invasive methods like transcranial magnetic stimulation (TMS) [2]. TMS uses a device that is placed on a patient's head and then creates electromagnetic fields to stimulate nerve cells (neurons) in the brain. In recent years, it has been successfully used to improve symptoms of mental health diseases such as depression and many other neurological and psychiatric disorders. Researchers, clinicians, and doctors are now focused on determining the optimal placement of TMS devices to target specific brain regions effectively.

4.2 Assessing Human Accuracy

For our study, we developed an interactive tool *TMS Simulator (2D)* (10) for users to handle a brain stimulation device in an abstracted 2D environment. The simulation interface allows users to control a 2D representation of an electromagnetic field to influence a section of brain's neuronal activity. In reality, the electromagnetic field is being generated by a stimulation device that a doctor hovers around a patient's head to treat brain diseases. The brain region is exemplified with a grid-like canvas of size $N \times N$, with each square reflecting a brain cell with an activity level of its enclosed neuron. The starting grid has some initial 'damages' with inactive neurons which exemplify the brain regions that need to be treated with the tool. Activity values can be manipulated using the circular tool representing the electromagnetic field. Users can move this tool with their mouse, adjust its size with a scroll, and initiate or pause treatment with a click. The application of the electric field on the brain cells happens in form of a Gaussian distribution similarly as a real-world electromagnetic field. The treatment clicks then change the activity value of the neurons under the circle, adjusting the colors of the squares, with colors ranging from inactive cells (dark grey) to optimal activity (white) to overly stimulated cells (red). Users are provided with a control legend and a color-coded legend for clarity. The simulation also evaluates the user's performance, calculating a score based on the brain's overall activity, and charts this score over time. The simulator has been accompanied by an interactive tutorial for users to learn and participate in the study without guided session moderator supervision.

Participants. The participant demographic for our user study, as detailed in Table 2, comprises of a sample with a diversity of educational backgrounds and a binary gender distribution. We recruited the participants via words of mouth and our collaboration network. The group included 13 individuals ($N = 13$), with a gender representation of 38.46% female (5 participants) and 61.54% male (8 participants). In terms of educational attainment, the participants are distributed across the spectrum of formal education levels: 7.69% (1 participant) have completed high school, 30.77% (4

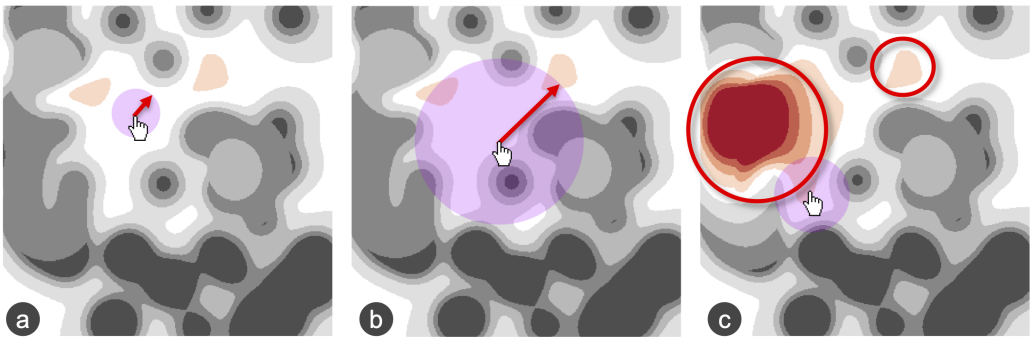


Fig. 11. **a)** 2D Grid of brain region with damaged cells in grey. The circular electric field (purple) has a small radius (red arrow). **b)** User can increase radius size of the electric field to a maximum of 1/4 of the grid by using the mouse scroll. **c)** User overexcited cells in a brain region with too strong electric field which is indicated with the red color.

Table 2. Participant demographic data from questionnaire

Gender		Female	Male	
		5 (38.46 %)	8 (61.54 %)	
Degree	High School	Bachelor	Master	PhD
	1 (7.69 %)	4 (30.77 %)	4 (30.77 %)	4 (30.77 %)

Table 3. User study scores, times and steps.

Metric	**General Users**	**Experts**	**Both Groups**
Average score [%]	96.60 ± std	92.44 ± std	95.32 ± 3.35
Average time [s]	388.44 ± std	216 ± std	335.38 ± 234.91
Average steps	5851.67 ± std	2125.25 ± std	4705.08 ± 3287.80

participants) hold a Bachelor's degree, another 30.77% (4 participants) have attained a Master's degree, and the remaining 30.77% (4 participants) possess a PhD. This distribution indicates a relatively balanced representation of educational backgrounds within the cohort, providing a broad perspective in the context of the user study. From the 13 participants, 4 were classified as *Experts* ($E1$ - $E4$), having a background in neuroscience or psychology, currently working in research on TMS or administering brain stimulation treatments to patients. The remaining 9 participants of our study, which we will call *General Users*, included mainly technical backgrounds in engineering or software development.

Study Protocol. We created a fully self-paced remote study hosted on a web-server. Participants received a link and could click through the study page by page, then followed by a link to a questionnaire. Sessions started with a tutorial to introduce the optimization problem for brain stimulation and to familiarize participants with the controls of the tool. Next, users engaged with the simulator to beat the score. For extra motivation we displayed the time passed from when the grid was first displayed. At the end of the simulator, we displayed a visualizations to let participants explore and reflect on their own performance, before going into a post-study questionnaire.

Table 4. Post-study questionnaire results

Question	Statements	General Users	Experts	Both Groups
Q1	Task goal clarity	4.56	4.25	4.46 ± 0.66
Q2	Ease of understanding color encoding	4.56	5	4.69 ± 0.48
Q3	Score interpretability	3.67	3.75	3.69 ± 1.25
Q4	Comfort in device manipulation/navigation	3.67	3.75	3.69 ± 1.11
Q5	Task completion ease	3.67	3	3.46 ± 1.20
Q6	Machine vs. human speed for task	4.56	3.75	4.31 ± 1.03
Q7	Machine vs. human score for task	4.67	4	4.46 ± 0.78
Q8	Trust in displayed score accuracy	4	3.75	3.92 ± 1.04
Q9	Simulator as 2D brain stimulation representation	3	3.5	3.15 ± 1.14

Post-study questionnaire. After the study, participants were asked to fill out a survey with 9 questions, to gathers qualitative feedback on participants' experience with the simulator, trust, its perceived strengths, potential areas for improvement, as well as overall satisfaction and user experience. The 9 questions were rated on a 5-point scale, where participants shared how much they agreed or disagreed with statements about the clarity of the task, how easy it was to understand the color codes for brain activity, and whether they were able to interpret the meaning of the displayed score. They also indicated how comfortable it was to use the simulated device, and how easy it was to complete the task. We were also interested in their assessment on whether they believed a computer could do the task quicker or better than a human, if they trusted the scores shown, and if the simulation was a good representation of the real-world 3D task.

4.3 Results

Table 3 shows the results from the user study. We logged times and scores into a database for post-processing. Each participant was assigned an anonymous session ID. Figure 12 shows participants' performance. Scores ranged from 84.92 % to 97.50 %, with the majority of participants achieving above 95 %. The *general* user group achieved an average score of 96.60 %. *Expert* users, while still proficient, had a slightly lower average score of 92.44 %, suggesting that the tool's challenges were robust across all levels of expertise. When combined, the overall average score for both groups was 95.32 % with a standard deviation of 3.35 %. The average time invested by users also differed notably between groups. Time spent in the simulation varied from 9 seconds up to 840 seconds (14 minutes), showing no strong correlation with neither scores or total steps taken, suggesting variable efficiency in tool use. General users took an average of 388.44 seconds, while experts completed tasks more rapidly, averaging 216 seconds, potentially reflecting familiarity and efficiency with such tasks. The combined average time for both groups was 335.38 seconds, but with a substantial standard deviation of 234.91 seconds, pointing to significant differences in individual completion times. Level of interaction, measured in total steps taken, varied extensively, with general users averaging 5,851.67 steps and experts 2,125.25 steps, indicating that experts navigated the tool with higher stability requiring fewer interactions. Steps taken as a measure of interaction also varied widely, from as few as 110 to as many as 10,841. This variability might indicate differences in user strategy, understanding, or the complexity of tasks they encountered. The overall average for both groups was 4,705.08 steps, with a large deviation once again highlighting the diversity in user approach.

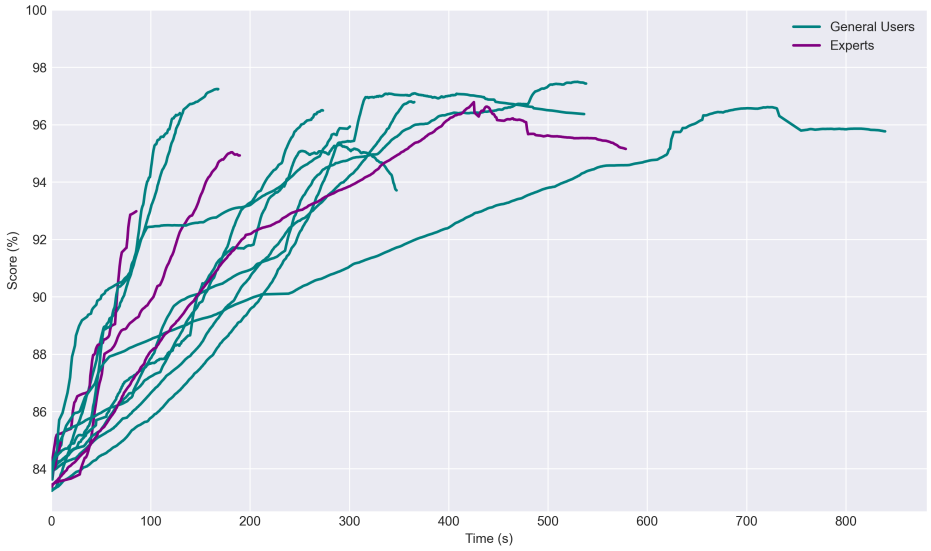


Fig. 12. **Scores over time for each participant:** Progression of scores over time for experts (in purple) versus general users (teal), illustrating the performance dynamics in user engagement with the 2D simulator tool. One expert curve is almost not visible because finishing up very fast. One general user took over 800 seconds, the line steady increasing due to very slow movements with the electric field.

The post-study questionnaire data as in Table 4 revealed insights into participant experiences with an unspecified tool, delineated between *general* and *expert* users. Participants rated their agreement with statements regarding various aspects of the tool on a likely Likert scale, with scores averaging between 3 and 5. Clarity of the task goal (Q1) and understanding of color encoding (Q2) received high ratings, indicating a clear design and intuitive interface, with averages across both groups being 4.46 (±0.66) and 4.69 (±0.48), respectively. Clarity on score meaning (Q3), comfort with device manipulation (Q4), and task completion ease (Q5) presented more moderate agreement, suggesting potential areas for improvement in user understanding and interface ergonomics. Notably, participants rated that the machine's speed (Q6) and score (Q7) compared to human performance for the task completion highly, averaging 4.31 (±1.03) and 4.46 (±0.78), respectively. This may indicate that participants believe that a program or machine will perform better than a human in this type of task. Trust in the accuracy of displayed scores (Q8) scored slightly lower, but still within a positive range, averaging at 3.92 (±1.04), hinting at slight reservations about the tool's reliability. Finally, the simulator's representation as a 2D brain stimulation (Q9) received the lowest average score of 3.15 (±1.14), which could signal a need for a more effective visual representation or user education regarding the simulation. The standard deviations suggest variability in participant responses, reflecting individual differences in perceptions and experiences with the tool.

5 RL ENVIRONMENT FOR BRAIN STIMULATION IN AUTORL X

Our research explores the application of RL to the previously presented brain stimulation problem through the creation of a specialized RL environment (*reference gym*). This simulated environment mimics essential aspects of brain stimulation procedures for educational and research applications. It is designed for ease of use, allowing individuals, irrespective of their RL expertise, to investigate and comprehend the nuances of brain stimulation techniques in a risk-free virtual setting. Similar to

how MuJoCo environments[6] provide simulation tasks for physical movements, our RL environment broadens the scope to healthcare, particularly the optimization of brain stimulation parameters and thereby optimal placement of an electromagnetic field on a brain region. This simulation can be essential for further devising personalized treatment strategies. With this algorithmic challenge, we had a variety of options how to implement the 2D TMS Simulator as a gym environment, specifically, since definitions of reward functions, action and observation spaces highly vary with decision of the developer. Our goal was to test this environment in AutoRL X and run it with a diverse number of agents.

5.1 Technical Implementation Details

Throughout this section we provide technical details on an exemplary gym implementation for our given TMS problem. We need to make decisions on how to model the state of the environemt, which is observed by an RL agent. A state is further assessed in a reward function, which the agent is tasked to optimize. In order to do so, the agent can propose an action, which is then applied to receive a new state, and so on. Typically, the procedure terminates after a certain number of steps or when a certain condition is met.

State/Observation. The state s of the gym is the representation of neural activity across the 2D grid, captured as a 2D vector. When returning the state in the OpenAI interface, the vector needs to be flattened to 1D: $s = \text{flatten}(\text{grid})$ where $\text{grid} \in \mathbb{R}^{N \times N}$ and $s \in \mathbb{R}^{N^2}$ with N being the side length of the grid.

Action. We can model the action space a as a 2-dimensional vector that represents the position of the TMS device in the grid ($device\_x$, $device\_y$) the location where the elecromagnetic field is applied:

$$a = [device\_x, device\_y]^T$$

where $device\_x, device\_y \in [0, N-1]$ are continuous values. If we wanted to further model the radius or the intensity of the field, we could do so by introducing additional actions analogously.

Transition/Step Dynamics. The transition dynamics are defined by the influence of the device on the grid. When an action is taken, it increases (stimulates) the values in the grid based on a Gaussian distribution centered around the action's location with a fixed or actionable radius R and intensity I:

$$\mathrm{s}_{x,y}^{t+1} = \mathrm{s}_{x,y}^{t} + \mathrm{OPT} \cdot \mathrm{I} \cdot \mathrm{Gaussian}(\mathrm{device\_x}, \mathrm{device\_y}, x, y, \mathrm{R}/2) \tag{1}$$

.

Reward Function. The reward function at any time step t could be modeled as the sum of the errors between the current grid values and the optimal activity level OPT, scaled by the maximum possible reward. Since reward demands the better the solution, the higher the value, we can simply take the negative. We further reward stimulation of understimulated neurons more than overstimulating already stimulated neuron by skewing the value distribution using an exponential function:

$$\mathrm{reward}(s) = -\frac{\sum_{x=1}^{N}\sum_{y=1}^{N}\left(\frac{\mathrm{grid}(x,y)}{\mathrm{OPT}} - 1\right)^2 \cdot \left(e^{\left(\frac{\mathrm{grid}(x,y)}{\mathrm{OPT}} - 1\right)} - 1\right)}{N^2 \cdot \mathrm{OPT}} \tag{2}$$

For visual clarity, Figure 13 shows the modeled function in comparison to a regular parabola.

[6]https://www.gymlibrary.dev/environments/mujoco/index.html

Termination. While typically the RL agent stops after a certain number of steps (horizon), we could additionally define a termination condition if all values in the grid reach above the optimal stimulation:

$$\text{done} = \min(\text{grid}) \geq \text{OPT}$$

Figure 13 further shows the source code implementation of the gym in Python.

5.2 AutoRL on the TMS Environment

We were interested in the actions that humans would take in optimizing their scores by comparing them to the RL agents' behaviors and actions. With the help of AutoRL X, we aim to analyze how the differently configured RL agents will perform in our defined reference environment to solve the same task as the study participants. We also hoped to receive similar graphs as in Figure 12, showing the steadily increasing curves of human performance to optimize the grid value. However, while training the agents with the TMS environment 1, our agents learned more back-and-forth depending on the hyperparameter tuners, selected reward function, and episodes.

The reference gym within the platform provides an observable environment where the behavior of the Reinforcement Learning (RL) agent can be monitored. This allows for real-time observation and analysis of the agent's interactions with the environment, offering tangible evidence of how different settings or configurations impact the agent's decisions. For instance, if an agent displays a propensity for selecting red or dark boxes within the environment, it could indicate that the scoring system may need adjustment to ensure the agent is not biased by certain visual features.

Additionally, AutoRL X serves as a valuable tool for conducting sanity checks to verify the fundamental correctness of the environment implementation — essentially, a form of debugging the created gyms. Observations might reveal that the actions chosen by the agent fall outside the expected range, leading to 'out of bounds' behaviors. Other observations we found are that agents might not necessarily move around the grid but sometimes also learn to get stuck in the corner and increase the score by executing the action on a single cell. Recognizing this enables developers to craft and compare different strategies for clipping or constraining actions, thus ensuring that the agent operates within the desired parameters. Figure 14 shows the alternative implementations we have added instead of AutoDOViz's matrix- and graph-based visualizations. Here, agents' behaviors in the 2D grid and trajectories are visualized over a single episode, enabled by the highly granular logging. At the beginning of step 1 in the first episode, the grid shows pre-defined damages, which are Gaussian distributed, and the agent has not yet interacted. In subsequent iterations, we can see how the agent moves to different positions in the grid, applying the electric field to the cells turning red when the value is over the optimal value (100 in our case). From the episode progress depicted in Figure 14, we can see that the agent has not yet developed a strategy to find the fastest and optimal trajectory to resolve the damaged cells in the grid. Finally, it is worth mentioning that our developed gym environment is also available as part of an open-source package to invite users to experiment and engage with the gym, to create a collaborative and exploratory approach to refine and enhance the gym, and to study RL models' behavior. The open-source nature grants community-driven development and the opportunity for diverse contributions that

can lead to innovative uses and improvements of gym setups through AutoRL X and beyond.

```
class MyEnv(BaseEnvironment):
    def __init__(self, obj_name="TMSSimulator2D", ...):
        super().__init__(...)
        self.gamma = 0.99
        self.horizon = 150
        self.N = 10
        self.OPT = 100
        self.n_damages = 10
        self.grid = self._init_grid()
        self.observation_space = Box(
            low=np.array([0.0] * (self.N ** 2)),
            high=np.array([self.OPT * 2.0] * (self.N ** 2)),
            shape=((self.N ** 2),))
        self.action_space = Box(
            low=np.array([0.0, 0.0]),
            high=np.array([self.N - 1.0, self.N - 1.0]),
            shape=(2,)
        )

    def _init_grid(self):
        grid = np.full((self.N, self.N), float(self.OPT))
        # damage grid
        np.random.seed(42)
        for _ in range(self.n_damages):
            kernel_x = np.random.randint(0, self.N)
            kernel_y = np.random.randint(0, self.N)
            for x in range(self.N):
                for y in range(self.N):
                    value = self.OPT * self._gauss(kernel_x,
                          kernel_y, x, y, 0.1 * self.N / 2)
                    grid[x][y] -= value
                    grid[x][y] = max(0, grid[x][y])
        return grid

    def _gauss(self, x0, y0, x, y, s):
        return math.exp(-((x - x0) ** 2) / (2 * s ** 2) - ((y - y0
              ) ** 2) / (2 * s ** 2))

    def step(self, action):
        device_x = int(abs(action[0]) % self.N)
        device_y = int(abs(action[1]) % self.N)
        r = 0.05 * self.N
        intensity = 0.1
        error = 0
        for x in range(self.N):
            for y in range(self.N):
                value = self.OPT * self._gauss(device_x, device_y,
                      x, y, r / 2) * intensity
                self.grid[x][y] += value
                d = -(self.grid[x][y] / self.OPT - 1) * 2
                error = error + d * (np.exp(d) - 1)
        score = -error / (self.N ** 2 * self.OPT)
        done = np.min(self.grid) >= self.OPT
        return self.grid.flatten(), score, done, {}

    def reset(self, initial_state=None):
        self.grid = self._init_grid()
        return self.grid.flatten()
```

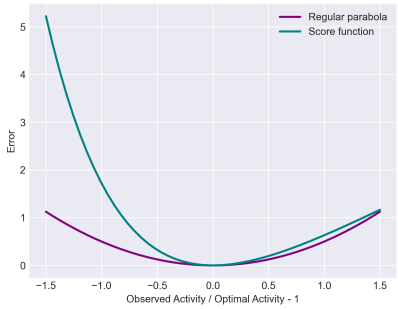
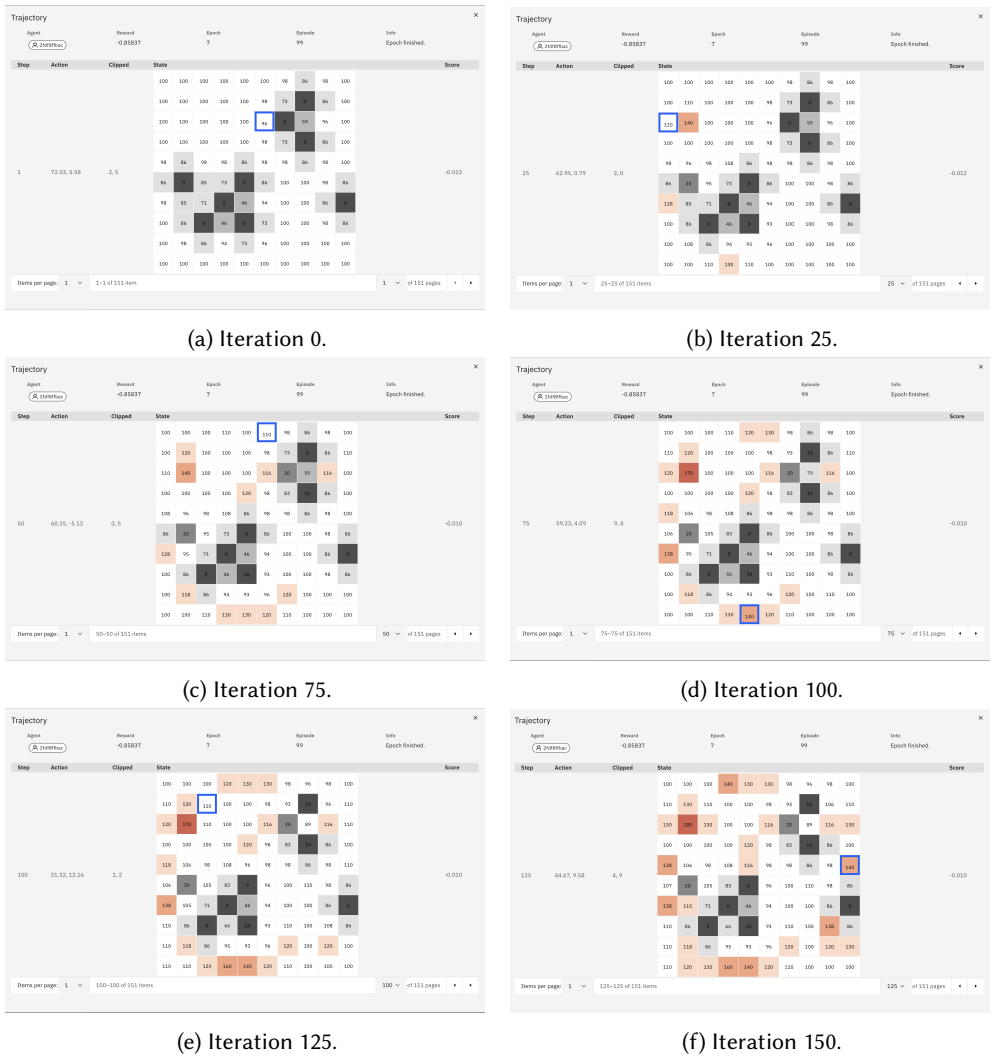


Fig. 13. Reward function that the agent receives: The teal function is showing the reward we pass to the agent when performing an action. We *punished* the agent more for leaving cells below the optimal value compared to over stimulating the cells of the grid with positive values.

6 DISCUSSION

Our development and exploration of a web-based user interface with visualizations demonstrates its effectiveness as an educational and problem-solving and debugging tool, substantially demystifying RL for a wider audience. Utilizing the AutoRL X platform, our study observed human and RL agents' behavior within a specially configured gym environment. While humans showed tendency to improve their performance steadily, RL agents vary their behavior based on hyperparameters,



(a) Iteration 0.

(b) Iteration 25.

(c) Iteration 75.

(d) Iteration 100.

(e) Iteration 125.

(f) Iteration 150.

Fig. 14. Trajectory of the agent for an initial epoch with 8 different iterations. The grid is initialized with cells of negative values (grey). The agent's current position on the grid is indicated by the blue box.

state and action space definitions and reward functions throughout and learning epochs. This was particularly evident in the RL agents' learning patterns, which sometimes resulted in repetitive or suboptimal actions, like getting stuck in a grid corner—a phenomenon that underscores the importance of precise environment design and agent investigation.

In the last section, we could demonstrate that the AutoRL X platform, besides its visual analytics potential derived from AutoDOViz, acts as a practical testing and debugging ground for custom-built RL environments, catering to users from novice to expert levels in RL. Through its detailed visual analytics, such as those presented in Figure 14, we could track the agents' interactions over time, offering insights into the agent's strategy development or lack thereof, as they interacted with the grid and adjusted to the simulated damages.

The responses from our post-study questionnaire show a consensus regarding the belief of the machine's superior speed and accuracy, which underscores the benefits of using automation in complex tasks like brain stimulation procedures. However, skepticism surrounding the accuracy of the displayed scores and lack of trust in the 2D simulator indicate a gap in the interface that necessitates more transparent feedback mechanisms.

Our goal, to develop a specialized RL environment for medical applications particularly simulating the complexities involved in brain stimulation, is an encouraging advancement. AutoRL X not only serves as an educational platform but also as a research instrument, offering a controlled environment to refine brain stimulation strategies. However, our findings indicate that an RL agent, although programmed to optimize scores, does not always match human strategies, highlighting areas for future visual analytics to advance agent and gym development. Furthermore, agent learning from human demonstrations is a promising approach addressed also more recently in the literature.

In conclusion, the user study and subsequent questionnaire provide valuable insights into human accuracy and perception of these tasks. The performance scores and varied time and interaction metrics demonstrate its adaptability and potential as a training and research platform.

6.1 Limitations

While AutoRL X makes a significant impact as an open-source platform for reinforcement learning, it is not without limitations. One of the challenges we encountered is integration and deployment of the platform, which presented severe compatibility issues across various backend frameworks due to different chipset architectures, and outdated dependencies compromising the heterogeneity of user environments. Despite flexible architecture decisions, seamless integration across user-specific configurations therefore remains an ongoing task. Furthermore, our interface strives to present information intuitively, yet the complexity of RL can make it challenging to distill information without sacrificing detail. As we continue to refine the user experience, we aim to tailor the information density to user preferences and expertise levels better. Next, despite having selected a diverse population, user study results would need to be confirmed via a larger sample before generalizing. The usability of AutoRL X was not directly tested, however, being the open-source continuation of AutoDOViz, design principles were informed by insights from previous extensive user studies. In light of this, we decided to forgo potentially redundant examination in favor of deploying a different type of user study to provide alternative insights into the process when working in highly domain specific environments.

6.2 Future Work

For the future trajectory of AutoRL X the remaining user requirements need to be addressed, such as collaborative features **R6** that enable real-time edits and comments for a cooperative learning environment. Additionally, embedding educational features **R11** like guided walk-throughs and interactive demonstrations as conducted in the simulator tutorial could significantly augment the learning curve for users new to RL. The TMS reference gym can be further explored to overcome the human accuracy provided in our user study. Furthermore, the potential for integrating AutoRL X into other tools or platforms should be investigated. This could lead to a more comprehensive ecosystem for RL and ML practitioners, promoting a seamless workflow across various tools. Despite the feedback and iterative improvements we have drawn from AutoDoViz, the need for ongoing refinement based on user engagement remains. Continuous user feedback is necessary for platform evolution, ensuring that AutoRL X not only meets latest user demands but also anticipates and adapts to quickly evolving ML landscape. This proactive approach to user-centric design and development will be crucial in maintaining the platform's relevance and effectiveness.

7 CONCLUSION

In this paper, we have presented AutoRL X, an open-source expansion of our previous work, AutoDOViz, which aims to contribute to better understanding and utilization of Reinforcement Learning (RL) in diverse domains. Our contributions encompass various facets that collectively advance the field of RL and highlight the critical role of visual analytics in promoting its understanding, trust and usage. Our foremost contribution lies in democratizing Automated RL technology with an open-source contribution. This ensures that our code is readily accessible to the community, fostering collaboration and innovation. The flexible architecture of AutoRL X allows seamless integration with various backend engines, making RL more approachable and adaptable for a broader audience. Building upon the insights and feedback garnered from interviews and user studies conducted during the development of AutoDOViz, we have tailored AutoRL X to address these identified user interface elements and incorporate additional features. This user-centric approach enhances the usability and personalization of RL agents, catered to the evolving needs of practitioners and researchers. Moreover, we have extended the applicability of our platform into the critical domain of healthcare. By creating a novel RL environment and a 2D simulator visualization component, we demonstrate the real-world potential of RL in optimizing complex healthcare challenges, such as optimizing brain stimulation device trajectories. Our user study, including experts from the healthcare field, provides valuable insights into the performance of RL compared to human decision-making, further solidifying the practicality of Automated RL. In summary, our work aims to leap forward to more intelligent user interfaces for RL, applying open-source technology and modern user interface design to bridge the gap between complex RL algorithms and tangible real-world problem-solving. By presenting AutoRL X, we hope to have contributed to the broader understanding of RL processes and emphasize the importance of visualization in enhancing RL trust and usage.

ACKNOWLEDGMENTS

The authors wish to thank the participants of the user studies and interviews in AutoDOViz which helped for our requirements analysis. We also wish to thank the experts of the TMS clinic and for their time and valuable input.

REFERENCES

[1] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. 2023. Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence. *Information Fusion* 99 (2023), 101805.

[2] Anthony T Barker, Reza Jalinous, and Ian L Freeston. 1985. Non-invasive magnetic stimulation of human motor cortex. *The Lancet* 325, 8437 (1985), 1106–1107.

[3] Andrew G Barto, Satinder Singh, Nuttapong Chentanez, et al. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*. ACM, Piscataway, NJ, 112–19.

[4] Saumyamani Bhardwaz and Rohan Godha. 2023. Svelte. js: The Most Loved Framework Today. In *2023 2nd International Conference for Innovation in Technology (INOCON)*. IEEE, New Delhi, India, 1–7.

[5] Ekaba Bisong. 2019. Google AutoML: cloud vision. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 581–598.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym.

[7] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2011. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction* 21 (2011), 137–180.

[8] Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. 2019. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters* 4, 2 (2019), 2007–2014.

[9] Po-Wen Chiu and Christina Bloebaum. 2008. Hyper-Radial Visualization (HRV) for decision-making in multi-objective optimization. In *46th AIAA Aerospace Sciences Meeting and Exhibit*. AIAA, 907.

[10] Piali Das, Nikita Ivkin, Tanya Bansal, Laurence Rouesnel, Philip Gautier, Zohar Karnin, Leo Dirac, Lakshmi Ramakrishnan, Andre Perunicic, Iaroslav Shcherbatyi, et al. 2020. Amazon SageMaker Autopilot: a white box AutoML solution at scale. In *Proceedings of the fourth international workshop on data management for end-to-end machine learning*. 1–7.

[11] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. 2021. MushroomRL: Simplifying Reinforcement Learning Research. *Journal of Machine Learning Research* 22, 131 (2021), 1–5. http://jmlr.org/papers/v22/18-056.html

[12] Shuby Deshpande, Benjamin Eysenbach, and Jeff Schneider. 2020. Interactive visualization for debugging rl.

[13] Aleksandra Faust, Anthony Francis, and Dar Mehta. 2019. Evolving rewards to automate reinforcement learning. *arXiv preprint arXiv:1905.07628* (2019).

[14] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. *Advances in neural information processing systems* 28 (2015).

[15] Jörg K. H. Franke, Gregor Köhler, André Biedenkapp, and Frank Hutter. 2020. Sample-Efficient Automated Deep Reinforcement Learning. *CoRR* abs/2009.01555 (2020). arXiv:2009.01555 https://arxiv.org/abs/2009.01555

[16] Loraine Franke and Daniel Haehn. 2020. Modern scientific visualizations on the web. *Informatics* 7, 4 (2020), 37.

[17] Claudio Gambella, Bissan Ghaddar, and Joe Naoum-Sawaya. 2021. Optimization problems for machine learning: A survey. *European Journal of Operational Research* 290, 3 (2021), 807–828.

[18] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. 2018. Visualizing and Understanding Atari Agents. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholm, Sweden, 1792–1801. https://proceedings.mlr.press/v80/greydanus18a.html

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[20] Wenbin He, Teng-Yok Lee, Jeroen van Baar, Kent Wittenburg, and Han-Wei Shen. 2020. DynamicsExplorer: Visual analytics for robot control tasks involving dynamics and LSTM-based control policies. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, Tianjin, China, 36–45.

[21] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622

[22] Martin Hirzel, Kiran Kate, Parikshit Ram, Avraham Shinnar, and Jason Tsay. 2022. Gradual AutoML using Lale. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4794–4795.

[23] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. 2018. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics* 25, 8 (2018), 2674–2693.

[24] Mingzhe Hu, Jiahan Zhang, Luke Matkovic, Tian Liu, and Xiaofeng Yang. 2023. Reinforcement learning in medical image analysis: Concepts, applications, challenges, and future directions. *Journal of Applied Clinical Medical Physics* 24, 2 (2023), e13898.

[25] Neil Hulbert, Sam Spillers, Brandon Francis, James Haines-Temons, Ken Gil Romero, Benjamin De Jager, Sam Wong, Kevin Flora, Bowei Huang, and Athirai A Irissappane. 2021. EasyRL: A Simple and Extensible Reinforcement Learning Framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 16041–16043.

[26] Theo Jaunet, Romain Vuillemot, and Christian Wolf. 2020. DRLViz: Understanding decisions and memory in deep reinforcement learning. *Computer Graphics Forum* 39, 3 (2020), 49–61.

[27] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1946–1956.

[28] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[29] James Max Kanter and Kalyan Veeramachaneni. 2015. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 1–10.

[30] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. 2016. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1304–1307.

[31] Dimitrios I Koutras, Athanasios C Kapoutsis, Angelos A Amanatiadis, and Elias B Kosmatopoulos. 2021. Marsexplorer: exploration of unknown terrains via deep reinforcement learning and procedurally generated environments. *Electronics* 10, 22 (2021), 2751.

[32] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. 2017. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327* (2017).

[33] Erin LeDell and Sebastien Poirier. 2020. H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, Vol. 2020.

[34] Yuxi Li. 2018. Deep Reinforcement Learning. *CoRR* abs/1810.06339 (2018). arXiv:1810.06339 http://arxiv.org/abs/1810.06339

[35] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[36] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. 2018. Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning*. PMLR, 561–591.

[37] Radu Marinescu, Tejaswini Pedapati, Long Vu, Paulito Palmes, Todd Mummert, Peter Kirchner, Dharmashankar Subramanian, Parikshit Ram, and Djallel Bouneffouf. 2022. Automated Decision Optimization with Reinforcement Learning. (2022).

[38] Aditi Mishra, Utkarsh Soni, Jinbin Huang, and Chris Bryan. 2022. Why? why not? when? visual explanations of agent behaviour in reinforcement learning. In *2022 IEEE 15th Pacific Visualization Symposium (PacificVis)*. IEEE, Tsukuba, Japan, 111–120.

[39] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[42] Deepak Mukunthu, Parashar Shah, and Wee Hyong Tok. 2019. *Practical Automated Machine Learning on Azure: Using Azure Machine Learning to Quickly Build AI Solutions*. O'Reilly Media.

[43] Marco Mussi, Davide Lombarda, Alberto Maria Metelli, Francesco Trovò, and Marcello Restelli. 2022. ARLO: A Framework for Automated Reinforcement Learning. https://doi.org/10.48550/arXiv.2205.10416 arXiv:2205.10416

[44] Hai Nguyen and Hung La. 2019. Review of Deep Reinforcement Learning for Robot Manipulation. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 590–595. https://doi.org/10.1109/IRC.2019.00120

[45] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016*. 485–492.

[46] Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, Yingjie Miao, Theresa Eimer, Baohe Zhang, Vu Nguyen, Roberto Calandra, Aleksandra Faust, et al. 2022. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research* 74 (2022), 517–568.

[47] Daniel Poh, Bryan Lim, Stefan Zohren, and Stephen Roberts. 2021. Building cross-sectional systematic strategies by learning to rank. *The Journal of Financial Data Science* 3, 2 (2021), 70–86.

[48] Emily Saldanha, Brenda Praggastis, Todd Billow, and Dustin Lockhart Arendt. 2019. ReLVis: Visual Analytics for Situational Awareness During Reinforcement Learning Experimentation.. In *EuroVis (Short Papers)*. IEEE, Porto, Portugal, 43–47.

[49] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[50] Wolfram Schultz, Peter Dayan, and P Read Montague. 1997. A neural substrate of prediction and reward. *Science* 275, 5306 (1997), 1593–1599.

[51] Syed Yousaf Shah, Dhaval Patel, Long Vu, Xuan-Hong Dang, Bei Chen, Peter Kirchner, Horst Samulowitz, David Wood, Gregory Bramble, Wesley M Gifford, et al. 2021. AutoAI-TS: AutoAI for time series forecasting. In *Proceedings of the 2021 International Conference on Management of Data*. 2584–2596.

[52] Gresa Shala, Sebastian Pineda Arango, André Biedenkapp, Frank Hutter, and Josif Grabocka. 2022. AutoRL-Bench 1.0. In *Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*. NeurIPS, New Orleans USA, Poster.

[53] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press, Cambridge, MA.

[54] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, IEEE, Vilamoura Portugal, 5026–5033.

[55] Vignesh Manoj Varier, Dhruv Kool Rajamani, Nathaniel Goldfarb, Farid Tavakkolmoghaddam, Adnan Munawar, and Gregory S Fischer. 2020. Collaborative suturing: A reinforcement learning approach to automate hand-off task in suturing for surgical robots. In *2020 29th IEEE international conference on robot and human interactive communication (RO-MAN)*. IEEE, 1380–1386.

[56] Dakuo Wang, Justin D Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. 2019. Human-AI collaboration in data science: Exploring data scientists' perceptions of automated AI. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–24.

[57] Junpeng Wang, Liang Gou, Han-Wei Shen, and Hao Yang. 2018. Dqnviz: A visual analytics approach to understand deep q-networks. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 288–298.

[58] Junpeng Wang, Wei Zhang, Hao Yang, Chin-Chia Michael Yeh, and Liang Wang. 2021. Visual analytics for rnn-based deep reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 4141–4155.

[59] Qianwen Wang, Yao Ming, Zhihua Jin, Qiaomu Shen, Dongyu Liu, Micah J Smith, Kalyan Veeramachaneni, and Huamin Qu. 2019. Atmseer: Increasing transparency and controllability in automated machine learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[60] Sen Wang, Daoyuan Jia, and Xinshuo Weng. 2018. Deep reinforcement learning for autonomous driving. *arXiv preprint arXiv:1811.11329* (2018).

[61] Daniel Karl I Weidele. 2019. Conditional parallel coordinates. In *2019 IEEE Visualization Conference (VIS)*. IEEE, Vancouver, Canada, 221–225.

[62] Daniel Karl I Weidele, Shazia Afzal, Abel N Valente, Cole Makuch, Owen Cornec, Long Vu, Dharmashankar Subramanian, Werner Geyer, Rahul Nair, Inge Vejsbjerg, et al. 2023. AutoDOViz: Human-Centered Automation for Decision Optimization. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*. ACM, Sydney, 664–680.

[63] Daniel Karl I Weidele, Justin D Weisz, Erick Oduor, Michael Muller, Josh Andres, Alexander Gray, and Dakuo Wang. 2020. AutoAIViz: opening the blackbox of automated artificial intelligence with conditional parallel coordinates. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. ACM, Cagliari, Italy, 308–312.

[64] Lindsay Wells and Tomasz Bednarz. 2021. Explainable ai and reinforcement learning—a systematic review of current approaches and trends. *Frontiers in artificial intelligence* 4 (2021), 550030.

[65] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. 2021. Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. In *Proceedings of the First ACM International Conference on AI in Finance* (New York, New York) *(ICAIF '20)*. Association for Computing Machinery, New York, NY, USA, Article 31, 8 pages. https://doi.org/10.1145/3383455.3422540

[66] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. 2021. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)* 55, 1 (2021), 1–36.

[67] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. 2016. Graying the black box: Understanding DQNs. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1899–1908. https://proceedings.mlr.press/v48/zahavy16.html

[68] Dongxia Zhang, Xiaoqing Han, and Chunyu Deng. 2018. Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems* 4, 3 (2018), 362–370. https://doi.org/10.17775/CSEEJPES.2018.00520

[69] Marc-André Zöller and Marco F Huber. 2021. Benchmark and survey of automated machine learning frameworks. *Journal of artificial intelligence research* 70 (2021), 409–472.

[70] Marc-André Zöller, Waldemar Titov, Thomas Schlegel, and Marco F. Huber. 2023. XAutoML: A Visual Analytics Tool for Understanding and Validating Automated Machine Learning. *ACM Trans. Interact. Intell. Syst.* (sep 2023). https://doi.org/10.1145/3625240 Just Accepted.

Received 6 November 2023; revised xx xx xxx; accepted xx.xx.xxxx