Diagnosing Faults in Deep Reinforcement Learning based Systems: Settings and Benchmarks

Blue Sky Ideas Track

Avraham Natan
Ben-Gurion University of the Negev
Beer Sheva, Israel
natanavr@post.bgu.ac.il

Roni Stern
Ben-Gurion University of the Negev
Beer Sheva, Israel
roni.stern@gmail.com

Meir Kalech
Ben-Gurion University of the Negev
Beer Sheva, Israel
kalech@bgu.ac.il

ABSTRACT

Deep Reinforcement Learning (DRL) is often used to generate control policies for autonomous agents. These policies are trained to control agents when they operate normally. Thus, unexpected faults may cause agents controlled using these policies to fail. When this occurs, it is important to understand and explain the root cause of such failures. In this paper, we define this diagnosis problem under different settings and assumptions. We also provide a benchmark suite for evaluating algorithms for solving this problem based on environments from AI Gym, a popular DRL framework.

KEYWORDS

Diagnosis; Autonomous Systems; Reinforcement Learning

ACM Reference Format:
Avraham Natan, Roni Stern, and Meir Kalech. 2026. Diagnosing Faults in Deep Reinforcement Learning based Systems: Settings and Benchmarks: Blue Sky Ideas Track. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 5 pages. https://doi.org/10.65109/NSTI8981

1 INTRODUCTION

Deep Reinforcement Learning (DRL) is widely employed for controlling systems where either no reliable model of the environment exists, or the available model is too complex to analyze effectively. Typical examples of such environments include autonomous cars and UAVs [3, 21], algorithmic trading [8] and game AI agents [18]. In such systems, an agent is situated in an environment where it can perform actions to collect reward and change its and the environment's state. The agent's goal is usually to interact with the environment in a way that maximizes the cumulative reward gathered by executing its actions. To achieve that, such systems assume available data collected through the agent's interaction with the environment or through simulations. Using this data, DRL approaches train *policies* that guide the agent to perform a series of actions that maximize its cumulative reward.

Such policies assume the agent operates normally. Normal operation means that the outcomes of the actions performed during policy training have similar probabilities of occurring during policy execution under similar environment states. Conversely, when an agent operates abnormally, some of its actions have unexpected effects, leading it to transition to different, often undesirable states during policy execution. Techniques in the DRL field, such as fault shielding [5, 22] and others [14], address those needs to some extent. However, with no understanding that this discrepancy is caused by faulty actions, those unexpected states are seen by the policy as the normal outcomes of its execution, and the policy continues to choose the next actions based on these faulty states, and so on. This may cause the agent to collect fewer rewards and fail to achieve its goals. In this work, we consider the problem of identifying the root cause of abnormal behavior in the execution of policy-guided systems, using a process called *diagnosis*. The result of such a diagnosis can then be used to guide more efficient repair processes and to complement the tasks of subsequent policy training with the gained knowledge of the new possible action outcomes. Moreover, such diagnostic results can enable learning of faulty action models, a complementary task to traditional learning of action models [28].

The problem of automated diagnosis has been extensively studied in the AI literature, and a variety of approaches have been proposed to address it. Diagnosis approaches include Data-Driven [6], Model-Based [15, 36], Rule-Based [23], and others [46]. Such approaches have been used for many problems ranging from industrial systems [10] to software [40]. To the best of our knowledge, utilizing diagnosis methods to find the root cause of a failure in systems that are guided by a DRL-generated policy has yet to be addressed. Diagnosis in such systems is challenging because they are often applied in domains where existing approaches to automated diagnosis are not directly applicable.

The contribution of this work lies in introducing a new research direction: the diagnosis of RL-based systems. In particular, **(1)** we present the problem of RL Diagnosis (RLDX). **(2)** We describe variants of the problem that depend on assumptions made about the environment, the policy, the faults, and the observations. **(3)** We provide benchmarks that include faulty executions of RL policies that can be used to test fault diagnosis algorithms.

2 BACKGROUND

Fault diagnosis focuses on identifying the root cause of faults that occur during the system's execution. Some notable approaches include Data-Driven [6, 44], Rule-Based [23], Model-Based [36] and Fault-Tree Analysis [46]. Diagnosis approaches address a wide variety of settings, including static and temporal systems, full and partial observability, and others. They are relevant to a wide variety of systems, from industrial [9, 10] to infrastructure [16, 35], UAVs [26, 39, 41], Software [1, 13, 40], and so on. Many of the approaches

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/NSTI8981

mentioned assume that the planned execution of the task is known before the actual execution, as in [7, 19]. Only a recent work [31] has tackled diagnosing faults in systems where the agents are guided by a policy. However, that work is based on many limiting assumptions, such as persistent faults and deterministic environments and policies. To that end, this paper lays some foundations and provides initial benchmarks for research in this direction.

Related to the problem of diagnosis is the problem of adaptability to failures, i.e., its ability to maintain or quickly return to operation in the presence of failures. Control approaches involve reacting in real-time to unexpected observations by dynamically adjusting the system's behavior to minimize disruptions [2, 20]. Replanning methods aim to modify the system's original plan when a failure or disturbance occurs by creating a new feasible plan to achieve the intended goal [25, 38, 48]. Robust planning aims to create plans that can handle failures without requiring significant real-time adjustments [4, 33]. Diagnosing failures is expected to help these methods by allowing them to focus on the root cause.

3 PROBLEM SETTINGS

We consider an autonomous agent situated in an environment that can be described as a *Markov Decision Process* (MDP). Formally:

Definition 1 (Markov Decision Process). *An MDP is defined by a tuple* $\langle S, A, P_a(s,s'), R_a(s,s')\rangle$ *where:*

- S *is a set of states.*
- A *is the set of actions the agent can perform.*
- $P_a(s,s') = Pr(s'|s,a)$ *is the probability of reaching state* s' *after the agent executes action* a *in state* s.
- $R_a(s,s')$ *is the immediate reward gained by executing action* a *from* s *and reaching* s'.

DRL algorithms, such as DQN [42], AC3 [27], and PPO [37], aim to return a *policy* that guides the agent in an MDP to choose actions that maximize its cumulative reward.

Definition 2 (Policy). *Given an MDP* $\langle S, A, P_a(s,s'), R_a(s,s')\rangle$, *a policy is a function* $\pi : S \to A$ *mapping a state* s *to the action the agent should take when in* s.[1]

DRL algorithms require interacting with the environment in order to output effective policies. Since this is not always possible, DRL algorithms often use a *simulator* of the environment.

Definition 3 (Simulator). *An environment simulator for an MDP* $\langle S, A, P_a(s,s'), R_a(s,s')\rangle$, *is a stochastic function* $\chi : S \times A \to S$ *such that* $Pr(\chi(s,a) = s') = P_a(s,s')$.

We use $\chi^i(s,\pi)$ to represent the sequential execution of i actions starting from state s and acting according to a policy π. That is, $\chi^i(s,\pi) = \chi(\chi^{i-1}(s,\pi),\pi)$ for $i > 1$ and $\chi^1(s,\pi) = \chi(s,\pi)$.

The main premise of this work is that during execution, some actions may *fail*, which means that executing them may transition the agent to a state very unlikely according to the transition function. To be able to detect and diagnose such failures, we assume a given sequence of observations $Obs = (o_1, \ldots, o_n)$, collected while the agent executed its policy, and an *observation consistency* function $consistent(Obs, \pi, \chi) \to [0,1]$, which quantifies how likely it is that Obs were collected while the agent executed actions using π and the resulting states were generated by the simulator χ. Here $consistent(Obs, \pi, \chi) = 0$ means Obs could not have been collected by the agent when choosing actions according to π and observing states according to χ. If the agents follow π then $consistent(Obs, \pi, \chi) = 0$ indicates at least one action has failed.

Definition 4 (RLDX). *Given a policy* π, *a simulator* χ, *a set of observations* $Obs = (o_1, \ldots, o_n)$, *and a threshold value* $T \in [0,1]$, *an RL Diagnosis (RLDX) problem arises when* $consistent(Obs, \pi, \chi) \leq T$. *A solution to an RLDX problem is a* diagnosis *that explains the observation and identifies the root causes of the failure.*

3.1 What is a Diagnosis in RLDX?

To define the notion of *diagnosis* more accurately, we adopt the consistency-based definition of a diagnosis as an assumption over the *fault modes* of the *system components* that is consistent with the observations. Next, we propose several alternatives to define fault modes and system components in the context of RLDX.

Fault modes in RLDX. It is common in MBD to distinguish between the two types of *fault models*: Weak Fault Model (WFM) and Strong Fault Model (SFM) [34]. WFM means the diagnoser only knows the expected behavior of system components. Consequently, every component in WFM has a single fault mode, indicating that it is abnormal. Thus, a diagnosis in WFM is an assumption about which subset of components is abnormal that is consistent with the observations. WFM can be a sufficient assumption for RLDX applications where removing or replacing faulty components must be done urgently, for example, to maintain a production process.

SFM assumes precise descriptions of how faults behave. Approaches that assume this often use mathematical or logical equations to precisely describe the effects of the fault over the system [45]. Such an assumption provides a layer of explainability to how the faulty components led to the overall failure — a thing that is useful for updating future policies or calculating compensation.

System components in RLDX. System models define the set of components that interact with one another. This definition can vary in granularity, which can affect the granularity of the resulting diagnoses. A possible definition is to treat each action execution in a specific state as a separate system component. In this case, a diagnosis would identify both the action type and the state in which it was executed. One may also define components more coarsely, considering only the action type regardless of the state in which it occurred. Naturally, the complexity of the diagnosis process is directly influenced by the chosen definition of system components.

3.2 Variants and Assumptions

RLDX problem instances may vary based on the assumptions they make about (1) the given policy, (2) the environment, (3) the types of faults that may occur, and (4) the observations we have.

Deterministic vs. stochastic environments. The standard definition of an MDP (Def. 1) suggests the environment is stochastic: the state transition function $P_a(s',s)$ defines the probability that an agent reaches state s' given its current state s and the chosen action a. Indeed, many real-world applications have elements of randomness

[1] This definition of a policy assumes deterministic action selection for ease of presentation. We discuss stochastic policies later.

which are naturally modeled as stochastic environments [17, 24, 43]. There are, however, important real-world applications that can be sufficiently represented as a deterministic environment [11, 12, 47]. Deterministic environments assume that the next state and reward are fully determined by the current state and the current action chosen by the agent. That is, the transition function $P_a(s', s)$ returns either one or zero for any action and pair of states.

Solving RLDX problems having a deterministic environment is expected to be easier since diagnosis algorithms can use the simulator to compute the exact next state given an observed state and action. If the next state is observed to be different, a diagnosis algorithm will immediately identify the performed action as part of any diagnosis. Conversely, diagnosis in stochastic environments can be significantly harder since the diagnosis algorithms may need to account for multiple outcomes of an observed action.

Deterministic vs. stochastic policies. Definition 2 implies that the agent's policy is deterministic. While deterministic policies are adequate in some cases, there are domains in which stochastic policies are preferred. A stochastic policy is a probability distribution over the actions the agent can take. When an agent executes a stochastic policy, it samples an action in each state from this distribution.

Solving RLDX problems having deterministic policies is expected to be easier since, given the policy and an observed state, we can uniquely identify the action performed in that state. Then, we can use the simulator to sample the possible next state and check if it matches the available observations, thus isolating the root cause. Diagnosis is still not trivial if observations are incomplete or noisy.

Solving RLDX problems with a stochastic policy is significantly more complex, since for every observed state, the diagnosis process may need to reason about multiple possible actions. Problems with both stochastic policies and stochastic environments are expected to be even more complex, potentially involving reasoning about multiple possible actions and the outcomes of each.

Intermittent vs. non-intermittent faults. The faults in a diagnosed system can be either intermittent or persistent (non-intermittent). Persistent faults persist until they are resolved or the task execution is complete. The persistence of non-intermittent faults makes them more predictable and easier to address using traditional monitoring or diagnostic systems such as model-based diagnosis. On the other hand, intermittent faults can occur unpredictably and may appear at varying intervals or when specific conditions are met. Examples include software bugs or systems with components periodically failing, for example, due to overheating. This makes detecting and reproducing intermittent faults more difficult. Methods such as model-based diagnosis will have to address two possibilities at every state - one where the fault occurred and one where it did not. In addition, if the observations appear normal, intermittent faults may lead to false negatives.

Full vs. partial observability of trajectories. Observability on the level of a trajectory focuses on whether all the states of a trajectory are observed (called full observability) or where some states are missing from the observed trajectory (called partial observability). In terms of the observation set *Obs* defined in Definition 4, partial observability on the trajectory level means that a subseries $(o_{i_1}, \ldots, o_{i_k})$ of the observations *Obs* might be missing.

In a fully observed setting, the diagnosis algorithm can access the complete series of states that make the trajectory. With such an assumption, diagnosis can be made easier since full observability provides diagnosis methods with more states to use for validation purposes during diagnosis generation. The setback of this assumption is that it is not always realistic. Domains that involve high communication costs, privacy-oriented components of the system, and limited observability capabilities may not always be able to provide a full series of states. When the trajectory is partially observable, the diagnosis engine does not have access to the entire series of states that make up the trajectory. Examples where such an assumption is valid include diagnosing space exploration tasks, robotics in dynamic environments, or GPS tracking. While diagnosis algorithms that assume partial trajectory observability can address more real-world cases, they are usually more complex, as they must infer the missing states, often leading to multiple diagnoses.

Full vs. partial observability of states. Partial observability on the state level means that some of the observed states in *Obs* may be observed partially. For example, if a state s is represented by v state variables, then when $u < v$ of those variables are observed, we say that the state is partially observable.

Assuming fully observed states means that every state that is observed provides full state information to the diagnosing algorithm. The advantage of this assumption is that there is no ambiguity regarding which state the system is currently in. Using this assumption, simpler and faster diagnosis algorithms can be employed, and the diagnoses returned can be more accurate, with fewer false negatives. The diagnosis community often makes this assumption, especially when the diagnosed system is in a controlled environment with reliable sensors [30]. However, in applications that are not controlled, such as space exploration or logistic chains, this assumption is not realistic. Assuming partial observability means that the diagnosing engine can access only parts of every state. This is especially useful when diagnosing systems distributively, where multiple diagnosis engines must diagnose part of the system [29, 32]. The limitations of this assumption are that the diagnosis algorithms may output more false negatives - diagnoses that are consistent only with the information provided by the observed states. This may require more complex diagnosis algorithms.

4 BENCHMARKS

This work provides a benchmark package of policy-guided executions for seven well-known DRL domains where faults were introduced during execution. For each domain, we run the environment using either a deterministic or stochastic policy while introducing faults using different predefined fault modes and varying fault probabilities. We next elaborate on the environments, policies, fault modes, and the benchmark generation steps.

4.1 Environments and Policies

The package includes the environments *Acrobot*, *CartPole*, *MountainCar*, *Taxi*, *FrozenLake*, *Breakout* and *Pong*, all of which are part of the Gymnasium benchmarks[2]. Figure 1 illustrates these environments. The benchmarks involve execution trajectories of determin-

[2]https://gymnasium.farama.org/

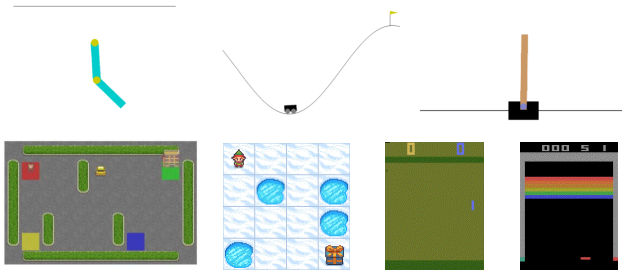

Figure 1: The AI Gym environments in our RLDX benchmark, from top to bottom and left to right: Acrobot, MountainCar, CartPole, Taxi, FrozenLake, Pong and Breakout.

istic and stochastic policies in environments with discrete sets of actions and deterministic and stochastic state transition functions. The details are shown in Table 1, and include the Gym environment name; the version of the environment; the number of actions an agent can perform; whether state variables are continuous or discrete; whether the environment is dynamic or static, where "dynamic" means the state may change even if the agent does not act, e.g., due to external forces operating over the agent; whether the state transitions function is deterministic or stochastic; and the DRL model used to train the policy we use for that environment.

Env. Name	Ver.	Act. Num.	State Type	Env. Type	State Trans.	Policy
Acrobot	v1	3	Cont.	Dyn.	Det.	PPO[10]
CartPole	v1	2	Cont.	Dyn.	Det.	PPO[10]
MountainCar	v0	3	Cont.	Dyn.	Det.	DQN[20]
Taxi	v3	6	Disc.	Stat.	Det.	PPO[30]
FrozenLake	v1	4	Disc.	Stat.	Sto.	PPO[40]
Breakout	v4	4	Disc.	Dyn.	Sto.	A2C[10]
PongNoFrameskip	v0	4	Disc.	Dyn.	Sto.	A2C[50]

Table 1: General statistics of our benchmark domains.

Policies. For each environment, we either trained a policy or used a publicly available one from the HuggingFace model repository. The algorithms used to train the policies were DQN [42], PPO [37], and A2C [27]. The policies are provided as part of the benchmarks.

Fault Modes. For each environment we define different fault modes. The fault modes are functions that, given an action provided by the policy, return another, possibly the same, but not always, actual action to be executed. For the CartPole environment, we define three fault modes due to the low number of agent actions. For each of the other environments, we define ten fault modes. The fault modes are specified in the input files we provide, which we used to generate the benchmarks package.

[10]Trained by the authors of this paper.
[20]https://huggingface.co/sb3/dqn-MountainCar-v0/tree/main
[30]https://huggingface.co/zap-thamm/PPO-Taxi-v3/tree/main
[40]https://huggingface.co/clement-w/PPO-FrozenLakeV1-rlclass/tree/main
[50]https://huggingface.co/mrm8488/a2c-PongNoFrameskip-v0/tree/main

4.2 Benchmarks Generation

For environment we defined an input file with the following inputs:

- domain_name - The name of the environment.
- model_name - the DRL model used to generate the policy.
- policy_type - stochastic or deterministic.
- seeds - the seeds used to determine the starting states.
- modelled_fault_modes - the set of fault modes we modeled.
- fault_probabilities - for a fault to occur in each step.
- instances - the instance numbers for each setting.

To create each environment benchmark, we:

(1) Set a seed that determines the starting state.
(2) Set the fault mode for the execution.
(3) Set the probability of a fault occurring in every step of the execution. Probability of 1.0 means persistent faults.
(4) Execute ten instances of each such combination, up to 200 steps. For each instance, we run the corresponding policy from a starting state determined by the seed and using the instance's corresponding fault mode and fault probability.

We generate a total of 10000 trajectories for the domains *Acrobot*, *MountainCar*, *Taxi*, *FrozenLake*, *Breakout* and *Pong*, and 3000 trajectories for *CartPole*. The trajectories are fully observed but can be masked to generate partially observable problems. We record the problem instances in Excel files. Among the recorded information is the domain (environment) name, the DRL model used to train the policy, and so on. The benchmarks are available on the following anonymous repository: https://github.com/avi-natan/benchmarks. The repository includes the benchmarks and the necessary policies and code to replicate or extend them.

5 CONCLUSION

Systems guided by DRL generated policies are rising in popularity, but there is not sufficient research done to address faults that can happen during the execution of such systems. To that end, we presented the problem of diagnosing faults in systems that are guided by policies generated by DRL models (RLDX). We described the different variants the problem can have — deterministic vs. stochastic policy, deterministic vs. stochastic environment, intermittent vs. non-intermittent faults, and full vs. partial observability on the level of the trajectory and states. We provided benchmarks of four deterministic domains (*Acrobot*, *MountainCar*, *CartPole*, *Taxi*), and of three stochastic domains (*FrozenLake*, *Breakout*, *Pong*). For each domain, we provided trajectories generated by the execution of a deterministic policy on a number of different starting states, with different rates of probability for faults to occur.

We expect this paper to spark interest in the diagnosis community and in the broader RL community. One research direction that can leverage the benchmarks provided in this work is the learning of fault modes. By doing so, future research can relax the assumption of candidate fault modes while still proposing methods for diagnosis that assume the presence of SFM. Another approach involves relaxing the assumptions of the existing policy and simulator, thereby utilizing observations obtained solely from the environment. Since we designed our execution code to be general, we believe such extensions will be easy to integrate into our proposed framework, should such a need arise.

ACKNOWLEDGMENTS

This research was funded by the Ministry of Science grants No. 0006908/1001802443 and 0008612/1001948158 and by ISF grant No. 1238/23 to Roni Stern.

REFERENCES

[1] Rui Abreu, Peter Zoeteweij, and Arjan JC Van Gemund. 2009. Spectrum-based multiple fault localization. In *2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 88–99.

[2] Ibrahim Ahmed, Marcos Quinones-Grueiro, and Gautam Biswas. 2023. Adaptive fault-tolerant control of octo-rotor UAV under motor faults in adverse wind conditions. In *AIAA SciTech 2023 Forum*. 2535.

[3] Szilárd Aradi. 2020. Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems* 23, 2 (2020), 740–759.

[4] Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou. 2020. Robust multi-agent path finding and executing. *Journal of Artificial Intelligence Research* 67 (2020), 549–579.

[5] Steven Carr, Nils Jansen, Sebastian Junges, and Ufuk Topcu. 2023. Safe reinforcement learning via shielding under partial observability. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 14748–14756.

[6] Xuewu Dai and Zhiwei Gao. 2013. From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis. *IEEE Transactions on Industrial Informatics* 9, 4 (2013), 2226–2238.

[7] Johan De Kleer and James Kurien. 2003. Fundamentals of model-based diagnosis. *IFAC Proceedings Volumes* 36, 5 (2003), 25–36.

[8] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28, 3 (2016), 653–664.

[9] Alexander Diedrich, Alexander Maier, and Oliver Niggemann. 2019. Model-based diagnosis of hybrid systems using satisfiability modulo theory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1452–1459.

[10] Alexander Diedrich, Stefan Windmann, and Oliver Niggemann. 2024. Solving industrial fault diagnosis problems with quantum computers. *Quantum Machine Intelligence* 6, 2 (2024), 66.

[11] Carlos Diuk, Alexander L Strehl, and Michael L Littman. 2006. A hierarchical approach to efficient reinforcement learning in deterministic domains. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. 313–319.

[12] Simon S Du, Jason D Lee, Gaurav Mahajan, and Ruosong Wang. 2020. Agnostic Q-learning with function approximation in deterministic systems: Near-optimal bounds on approximation error and sample complexity. *Advances in Neural Information Processing Systems* 33 (2020), 22327–22337.

[13] Amir Elmishali, Bruno Sotto-Mayor, Inbal Roshanski, Amit Sultan, and Meir Kalech. 2021. Beirut: Repository mining for defect prediction. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 47–56.

[14] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[15] Alvin I Goldman and Chandra Sekhar Sripada. 2005. Simulationist models of face-based emotion recognition. *Cognition* 94, 3 (2005), 193–213.

[16] Ariel Gorenstein and Meir Kalech. 2022. Predictive maintenance for critical infrastructure. *Expert Systems with Applications* 210 (2022), 118413.

[17] Shao-Ming Hung and Sidney N Givigi. 2016. A Q-learning approach to flocking with UAVs in a stochastic environment. *IEEE transactions on cybernetics* 47, 1 (2016), 186–197.

[18] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. 2019. Deep learning for video game playing. *IEEE Transactions on Games* 12, 1 (2019), 1–20.

[19] Meir Kalech and Avraham Natan. 2022. Model-based diagnosis of multi-agent systems: A survey. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 12334–12341.

[20] Benjamin Kelm, Stephan Myschik, and Oliver Niggemann. 2023. Control Reconfiguration of CPS via Online Identification Using Sparse Regression (SINDYc). In *International Conference on Machine Learning For Cyber-Physical Systems*. Springer, 51–63.

[21] William Koch, Renato Mancuso, Richard West, and Azer Bestavros. 2019. Reinforcement learning for UAV attitude control. *ACM Transactions on Cyber-Physical Systems* 3, 2 (2019), 1–21.

[22] Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler, and Roderick Bloem. 2023. Online shielding for reinforcement learning. *Innovations in Systems and Software Engineering* 19, 4 (2023), 379–394.

[23] Mark A Kramer and BL Palowitch Jr. 1987. A rule-based approach to fault diagnosis using the signed directed graph. *AIChE journal* 33, 7 (1987), 1067–1078.

[24] Hui Li, Xuejun Liao, and Lawrence Carin. 2009. Multi-task Reinforcement Learning in Partially Observable Stochastic Environments. *Journal of Machine Learning Research* 10, 5 (2009).

[25] Jiaoyang Li, Zhe Chen, Yi Zheng, Shao-Hung Chan, Daniel Harabor, Peter J Stuckey, Hang Ma, and Sven Koenig. 2021. Scalable rail planning and replanning: Winning the 2020 flatland challenge. In *Proceedings of the international conference on automated planning and scheduling*, Vol. 31. 477–485.

[26] Shaojun Liang, Shirong Zhang, Yuping Huang, Xing Zheng, Jian Cheng, and Sisi Wu. 2022. Data-driven fault diagnosis of FW-UAVs with consideration of multiple operation conditions. *ISA transactions* 126 (2022), 472–485.

[27] Volodymyr Mnih. 2016. Asynchronous Methods for Deep Reinforcement Learning. *arXiv preprint arXiv:1602.01783* (2016).

[28] Argaman Mordoch, Brendan Juba, and Roni Stern. 2023. Learning safe numeric action models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 12079–12086.

[29] Avraham Natan and Meir Kalech. 2022. Privacy-aware distributed diagnosis of multi-agent plans. *Expert Systems with Applications* 192 (2022), 116313.

[30] Avraham Natan, Roni Stern, and Meir Kalech. 2023. Blame Attribution for Multi-Agent Path Finding Execution Failures. In *ECAI 2023*. IOS Press, 1763–1770.

[31] Avraham Natan, Roni Stern, and Meir Kalech. 2024. Diagnosing Non-Intermittent Anomalies in Reinforcement Learning Policy Executions (Short Paper). In *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 23–1.

[32] Avraham Natan, Roni Stern, Meir Kalech, William Yeoh, and Tran Cao Son. 2024. Diagnosing Multi-Agent STRIPS Plans. In *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 8–1.

[33] Michal Nekvinda and Roman Barták. 2021. Contingent planning for robust multi-agent path finding. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 487–492.

[34] Chris Preist, Kave Eshghi, and Bruno Bertolino. 1994. Consistency-based and abductive diagnoses as generalised stable models. *Annals of Mathematics and Artificial Intelligence* 11 (1994), 51–74.

[35] Marcos Quiñones-Grueiro, Marlon Ares Milián, Maibeth Sánchez Rivero, Antônio J Silva Neto, and Orestes Llanes-Santiago. 2021. Robust leak localization in water distribution networks using computational intelligence. *Neurocomputing* 438 (2021), 195–208.

[36] Raymond Reiter. 1987. A theory of diagnosis from first principles. *Artificial intelligence* 32, 1 (1987), 57–95.

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

[38] Tomer Shahar, Shashank Shekhar, Dor Atzmon, Abdallah Saffidine, Brendan Juba, and Roni Stern. 2021. Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research* 70 (2021), 923–954.

[39] Jia Song, Weize Shang, Shaojie Ai, and Kai Zhao. 2022. Model and data-driven combination: a fault diagnosis and localization method for unknown fault size of quadrotor UAV actuator based on extended state observer and deep forest. *Sensors* 22, 19 (2022), 7355.

[40] Bruno Sotto-Mayor and Meir Kalech. 2021. Cross-project smell-based defect prediction. *Soft Computing* 25, 22 (2021), 14171–14181.

[41] Ido Tam, Meir Kalech, Lior Rokach, Eyal Madar, Jacob Bortman, and Renata Klein. 2020. Probability-based algorithm for bearing diagnosis with untrained spall sizes. *Sensors* 20, 5 (2020), 1298.

[42] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[43] Yixuan Wang, Simon Sinong Zhan, Ruochen Jiao, Zhilu Wang, Wanxin Jin, Zhuoran Yang, Zhaoran Wang, Chao Huang, and Qi Zhu. 2023. Enforcing hard constraints with soft barriers: Safe reinforcement learning in unknown stochastic environments. In *International Conference on Machine Learning*. PMLR, 36593–36604.

[44] Long Wen, Xinyu Li, Liang Gao, and Yuyan Zhang. 2017. A new convolutional neural network-based data-driven fault diagnosis method. *IEEE Transactions on Industrial Electronics* 65, 7 (2017), 5990–5998.

[45] Franz Wotawa. 2014. Failure Mode and Effect Analysis for Abductive Diagnosis.. In *DARe@ ECAI*.

[46] Mohammad Yazdi, Javad Mohammadpour, He Li, Hong-Zhong Huang, Esmaeil Zarei, Reza Ghasemi Pirbalouti, and Sidum Adumene. 2023. Fault tree analysis improvements: A bibliometric analysis and literature review. *Qual. Reliab. Eng. Int.* 39, 5 (2023), 1639–1659. https://doi.org/10.1002/QRE.3271

[47] Wanpeng Zhang, Xiaoyan Cao, Yao Yao, Zhicheng An, Xi Xiao, and Dijun Luo. 2021. Robust model-based reinforcement learning for autonomous greenhouse control. In *Asian Conference on Machine Learning*. PMLR, 1208–1223.

[48] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. 2021. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics* 37, 6 (2021), 1992–2009.