Toward a Unified Benchmark and Taxonomy of Stochastic Environments

Aryan Amit Barsainyan
ophv91@visitor.nus.edu.sg
aryan.barsainyan@gmail.com

Jing Yu Lim
jy_lim@comp.nus.edu.sg

Dianbo Liu
dianbo@nus.edu.sg

Abstract

Reinforcement Learning (RL) agents have achieved strong results on benchmarks such as Atari100k, yet they remain limited in robustness to real-world conditions. Model-Based RL approaches that rely on learned World Models often struggle in environments with true stochasticity and partial observability, despite their theoretical grounding in POMDPs. Current benchmarks rarely capture these challenges, focusing instead on deterministic or overly simplified settings, and the lack of a clear taxonomy of stochasticity further hampers systematic evaluation. To address this gap, we introduce STORI (STOchastic-ataRI), a benchmark that incorporates diverse stochastic effects and enables rigorous assessment of RL methods under varied forms of uncertainty. In addition, we propose a taxonomy of stochasticity in RL environments, providing a unified framework for analyzing and comparing approaches.

1 Introduction

Autonomous driving, dialogue systems, or robot navigation agents often have to operate in environments where the same action can lead to different outcomes due to noise, partial observability, or stochastic dynamics [Antonoglou et al., 2022, Paster et al., 2022].

Recent advances in Model-Based Reinforcement Learning have achieved strong results across a diverse range of environments [Hafner et al., 2024, Zhang et al., 2023, Alonso et al., 2024]. Their ability to handle partial observability is often demonstrated using pixel-based environments. However, in many of these cases the visual observations are nearly lossless representations of the underlying state, and the benchmarks themselves are typically fully deterministic [Tassa et al., 2018, Ye et al., 2021].

This mismatch between deterministic benchmarks with state-complete image observations and the stochastic, partially observable nature of real-world settings poses a significant barrier to transferring RL advances beyond simulation. Achieving robustness in stochastic environments requires benchmarks that capture diverse forms of uncertainty, yet most existing suites emphasize simplified or deterministic conditions.

To address this gap, we introduce STORI (STOchastic-ataRI), a benchmark that systematically incorporates diverse stochastic effects and enables rigorous evaluation of RL techniques under different forms of uncertainty into Atari environments. Alongside, we propose an updated taxonomy of stochasticity in RL environments, providing a unified framework for analyzing and comparing approaches.

Table 1: Environment stochasticity taxonomy

Type ID	Stochasticity Type	Sub Type
0	Deterministic	NA
1	Intrinsic	Action Dependent
2	Intrinsic	Action Independent - Random
3	Intrinsic	Action Independent - Concept Drift
4	Partially Observed	Representation Learning
5	Partially Observed	Missing State Variable(s)

2 Related Work

The limitations of RL approaches in stochastic environments are often hidden because most widely used benchmarks, such as Atari games in the Arcade Learning Environment (ALE) [Bellemare et al., 2013], are deterministic or nearly deterministic [Paster et al., 2022]. Several approaches have been proposed to introduce stochasticity into the dynamics of the ALE, aiming to encourage and evaluate robustness in RL agents. Examples include injecting stochasticity [Hausknecht and Stone, 2015], random no-ops [Mnih et al., 2015], human starts [Nair et al., 2015], and random frame skips [Brockman et al., 2016].

This issue was further addressed by Machado et al. [2018], who introduced sticky actions as a general stochasticity parameter for the Atari benchmark. Sticky actions are free from researcher bias, do not interfere with the agent's action selection, and discourage agents from relying on memorization. While this represents an important advance, it remains limited in scope.

However, while influential, this mechanism captures only a narrow form of environmental stochasticity. Robustness to sticky actions does not imply robustness to other types of uncertainty, such as noisy observations, variable dynamics, or non-stationarity. STORI (STOchastic-ataRI) addresses this need by extending ALE with diverse, controllable sources of stochasticity which enables fine-grained evaluation of RL agents across multiple dimensions of uncertainty.

The taxonomy of stochasticity in STORI builds on and extends the classification outlined by Antonoglou et al. [2022], which emphasize that many environments are inherently stochastic and cannot be well approximated by deterministic models. They further note that partial observability can make environments appear stochastic when aliased states cannot be disambiguated, and that large, complex environments may similarly appear stochastic to agents with limited capacity.

3 STORI - Stochastic Atari Benchmark

In this section, we describe the taxonomy used to classify benchmark environments for different types of stochasticity. Standard Atari games such as Breakout and Boxing have been modified to allow fine-grain control of these stochasticity, enabling systematic evaluation of the robustness and adaptability of RL algorithms under diverse uncertainties, while providing a consistent framework for comparison across tasks.

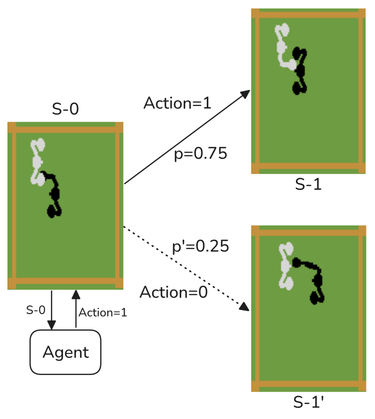
Table 1 presents the taxonomy of stochasticity, listing each type, its corresponding subtype, and the associated ID. Each type is explained in the following sections.

Type 0: Deterministic Environment

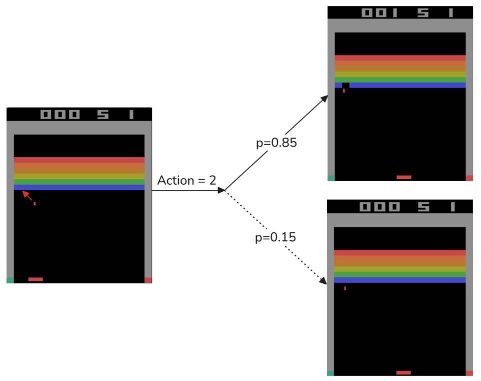
Deterministic environments are those in which the next state is fully determined by the current state and the action taken. The state is completely observable and there is no randomness in the state transitions or rewards, meaning that the outcome of any action is predictable.

Type 1: Intrinsic Stochastic Environment (Action Dependent)

In environments with action-dependent intrinsic stochasticity, the environment may, by default, replace the agent's chosen action with a random one. For instance, in $sticky\_action$ [Machado et al.,



(a) Type 1



(b) Type 2

Figure 1: **Left**: Type 1 stochastic state transition dynamics in Atari Boxing environment with action-dependent intrinsic stochasticity. There is a probability of 0.25 that a random action from action space is executed instead of action=1 predicted by the agent. **Right**: Type 2 Action Independent - Random modeled in Atari Breakout, where the paddle is moved to the right while the ball is on a trajectory to hit a block. In this case, there is a 0.15 probability that the ball bounces back without destroying the block or yielding any reward.

2018] scenarios, the environment can repeat the previous action with some probability. This results in varied outcomes even from the same state, with the stochastic effects limited to the state variables that can be influenced by the agent's actions. An example of action-dependent intrinsic stochasticity with Atari Boxing can be seen in figure 1a.

Type 2: Intrinsic Stochastic Environment (Action Independent - Random)

In action-independent random stochastic environments, randomness arises independently of the agent's choices and affects state variables outside the agent's direct control. This stochasticity, often due to external factors or inherent environmental noise, means that even with complete knowledge of the environment and carefully chosen actions, the next state cannot be predicted with certainty.

Figure 1b illustrates an example of how this type of stochasticity can be modeled in Atari Breakout, where the paddle is moved to the right while the ball is on a trajectory to hit a block. In this case, there is a 0.15 probability that the ball bounces back without destroying the block or yielding any reward. Notably, this stochastic behavior is independent of the action of moving the paddle to the right.

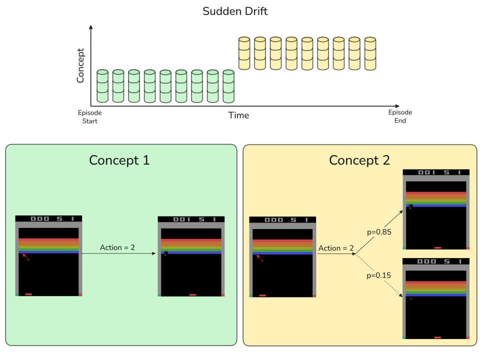
Type 3: Intrinsic Stochastic Environment (Action-Independent Concept Drift)

Environments with action-independent concept drift can change over time independently of the agent's actions, a phenomenon known as concept drift [Lu et al., 2018], which can generally be categorized into three types according to how the drift unfolds over time. In *sudden drift*, the environment undergoes abrupt changes, forcing the agent to quickly adapt to new dynamics. In *gradual or incremental drift*, the transition to new dynamics occurs slowly over time, requiring the agent to continuously adjust its policy. Finally, in *recurring drift*, previously observed dynamics reappear in a cyclical or context-dependent manner, making long-term adaptation more challenging. Learning in such environments demands flexibility and the ability to detect and respond to changes.

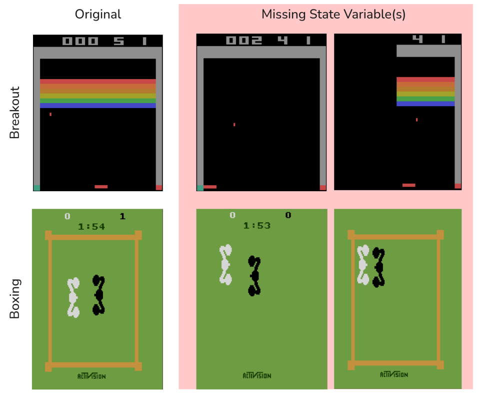
In the case of Atari, most games have intrinsic incremental drift. As the agent levels up in the game, the difficulty of the game also increases. With a more fine-grain control over concept drift, other types of drift can also be achieved in Atari games as shown in figure 2a.

Type 4: Partially Observed Environment (Representation Learning)

In partially observed environments, the agent does not have access to the full state information. When the state variables are represented differently from the true underlying environment, the agent must



(a) Type 3



(b) Type 5

Figure 2: **Left**: Type 3 Action Independent - Concept Drift modeled in Atari Breakout, where the concept suddenly changes from default Breakout dynamics (concept 1) to type 2 style environment dynamics (concept 2), within the same episode. **Right**: Type 5 - partially observed environments with missing state variable(s) using Atari Breakout and Boxing. Examples include invisible blocks or a partially hidden screen in Breakout and a hidden boxing ring or concealed clock and score information in Boxing.

infer hidden information or learn an appropriate representation. This increases the complexity of decision making, since the agent must rely on approximate observations.

A typical example is the default Atari setting, where the agent perceives only the screen image produced by the emulator after each action. These images are designed to approximate the true state, but they do not capture it fully. To enrich the observation, many implementations use a 4-frame skip with aggregation, which allows the agent to infer additional information, such as motion or rate of change over time, that is not apparent from a single frame.

Type 5: Partially Observed Environment (Missing State Variable(s))

An important subclass of partially observed environments arises when information about certain state variables is missing, leaving the agent unable to observe critical aspects of the environment. This lack of information demands strategies that can manage uncertainty and make robust decisions despite gaps in perception. Such environments are common in real-world scenarios where sensors are limited, noisy, or unreliable.

Figure 2b illustrates type 5 environments using Atari Breakout and Boxing. In Breakout, examples include invisible blocks or a partially hidden screen, while in Boxing, examples include a hidden boxing ring or concealed clock and score information.

4 Experiments

4.1 Experimental Setup

We employed the STORI benchmark to systematically evaluate the performance of recent model-based reinforcement learning algorithms, including DreamerV3 [Hafner et al., 2024] and STORM [Zhang et al., 2023]. DreamerV3 is a model-based reinforcement learning method featuring a learned world model paired with actor-critic architecture. It stands out for being robust across diverse tasks—all using fixed hyperparameters, with stability achieved via normalization, balancing techniques, and transformations. It also scales effectively with model size. STORM (Stochastic Transformer-based wORld Model) is likewise model-based RL, but employs a Transformer backbone with stochastic variational modeling. This design leverages strong sequence modeling and generation capabilities while embracing randomness for robustness and efficient learning. Although the STORI benchmark can, in principle, be extended to model-free approaches, doing so typically required substantially more computational resources to achieve statistically meaningful results, hence it was not included in this work.

Atari Breakout and Boxing were selected as the starting point for this benchmark because they are considered Agent-Optimal environments when stochasticity is not introduced [Lim et al., 2025]. This makes them suitable baselines for comparison once different types of stochasticity are applied. The two games also provide contrast in their action spaces, with Breakout offering 4 discrete actions and Boxing offering 18.

The design space of the STORI benchmark is very large, as each stochasticity type can be instantiated with several modes and a range of probability values. Exploring all possible combinations would be computationally prohibitive, and thus only a limited subset of configurations was feasible for this study. In this work, we therefore evaluate algorithms on a carefully chosen a specific instantiation for every type of stochasticity and the detailed settings can be found in the Appendix A.2 section.

Each algorithm was trained for 100K steps on the default environment versions as baselines and a range of modified environments of Breakout and Boxing, for 2 different seeds. Each run was evaluated on 100 episodes and the mean return across all 100 episodes was reported.

4.2 Results

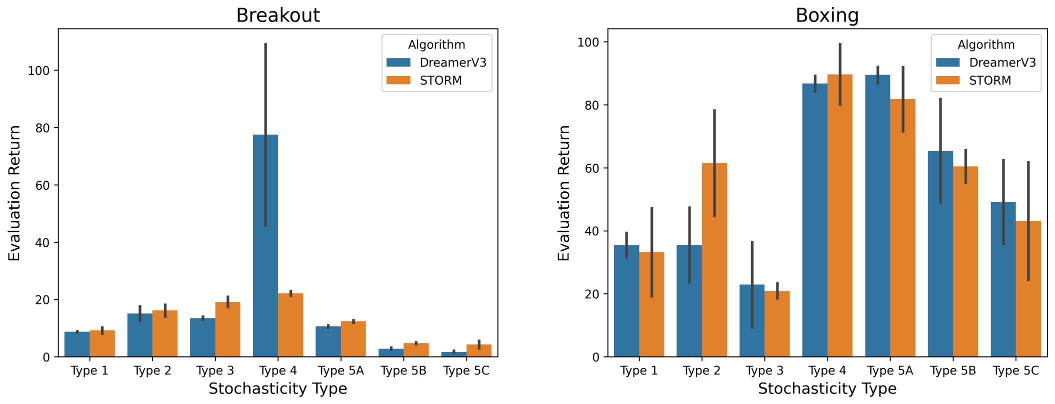


Figure 3: Comparison of DreamerV3 and STORM performance across different stochasticity types, with Type 4 serving as the baseline (without any added stochasticity).

The results from figure 4 indicate that the introduction of stochasticity led to a marked decline in performance across both Atari Breakout and Boxing when compared to the default (Type 4) setting. This outcome highlights the increased difficulty posed by additional sources of uncertainty, which fundamentally alter the learning dynamics of reinforcement learning algorithms.

In Breakout, DreamerV3 demonstrated clear superiority over STORM in the default baseline environment. However, this advantage diminished once stochasticity was introduced. In fact, STORM exhibited slightly stronger performance across all stochasticity types, suggesting a greater degree of robustness to unpredictable environment dynamics. In Boxing, the performance gap between the baseline and stochastic variants was less pronounced than in Breakout, though performance still declined consistently under stochastic conditions. DreamerV3 outperformed STORM across several stochasticity types, while STORM maintained a slight advantage in the default baseline.

Although Boxing has a larger action space than Breakout, its performance under stochasticity degrades less severely. This can be attributed to action redundancy, as many of Boxing's 18 actions are functionally similar, meaning stochastic perturbations often produce behavior close to the intended action. By contrast, Breakout's small action set is highly sensitive where an incorrect LEFT or RIGHT can immediately result in failure. Another important difference lies in how the environments handle recovery from mistakes. In Boxing, agents can retreat or reposition, creating opportunities to reset their strategy when stochasticity leads to unexpected outcomes. This recovery buffer mitigates the long-term impact of uncertainty, whereas Breakout provides virtually no margin for error. Taken together, this potentially highlights that the STORM has an advantage in handling high-stakes uncertainty compared to DreamerV3.

A notable case is the Type 5B Boxing environment, where the right vertical half of the screen is randomly hidden for 75% of the episode (p=0.75). In this setting, both algorithms discovered an effective policy that exploited the environment's structure where keeping the opponent confined to the visible left half of the screen, either by maintaining distance or by maneuvering around to force the opponent into that region. This illustrated how agents can adapt by leveraging spatial constraints in partially observed environments, turning limited visibility into a manageable challenge rather than a prohibitive handicap.

Interestingly, for the Type 5A setting in boxing, where the score and game clock information are permanently hidden from the screen, DreamerV3 outperformed its own performance on the default baseline environment. This suggests that the absence of certain state variables does not necessarily degrade performance and, in some cases, may even simplify the agent's representation learning by reducing distractions from non-essential information.

When examining the concept drift results, we find contrasting behaviors between Boxing and Breakout, with no clear winner between the two algorithms. In Boxing (Type 3), episodes begin in the default setting and transition abruptly to the Type 5C environment after 300 steps. Here, overall performance is notably worse than in the standalone Type 5C environment. In contrast, in Breakout (Type 3), where episodes start in the default setting and shift to Type 5A after 300 steps, both algorithms achieve slightly better performance than in the standalone Type 5A environment.

4.3 Limitations

It is important to emphasize that each stochasticity type modifies the underlying environment in a distinct way. As a result, environments with different forms of stochasticity are not directly comparable to one another in many cases for example Type 2 experiment in Breakout where the hits get stochastically canceled thereby reducing potential for high reward even with optimal performance from agent. Instead, a better point of reference is a comparison of algorithmic performance within each stochastic environment.

Additionally, while the large design space of the STORI benchmark provides valuable flexibility and control, it also introduces the risk of researcher bias when benchmarking algorithms. Careful task selection and transparent reporting are therefore essential to ensure fair and reliable comparisons.

In this work, computational constraints restricted experimental runs to only two seeds, which limits the statistical reliability of the results and one needs be cautious before drawing strong conclusions about algorithmic performance. Furthermore, these constraints also prevented evaluation of state-of-the-art algorithms such as DIAMOND [Alonso et al., 2024], JEDI [Lim et al., 2025], as well as several model-free approaches.

5 Conclusion

Our results underscore how different forms of stochasticity fundamentally impact the relative performance of different reinforcement learning algorithms. It also highlights cases where environments with small, sensitive action spaces amplify the impact of uncertainty, while those with redundant actions or recovery opportunities mitigate performance loss. Partial observability and selective state removal can even simplify learning, revealing that more information is not always better.

The STORI benchmark provides a flexible and configurable framework to explore these dynamics systematically. By varying stochasticity types and environment configurations, it allows researchers to probe algorithmic robustness, adaptability, and strategy formation under uncertainty, offering analogues to real-world scenarios with similar stochastic characteristics.

Overall, our work highlights that beyond raw performance metrics, stochastic benchmarks are a powerful tool for understanding how algorithms respond to uncertainty, guiding the design of more resilient and adaptable reinforcement learning systems.

Acknowledgments and Disclosure of Funding

References

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=X6D9bAHhBQ1`.

Keiran Paster, Sheila A. McIlraith, and Jimmy Ba. You can't count on luck: why decision transformers and rvs fail in stochastic environments. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024. URL `https://arxiv.org/abs/2301.04104`.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. STORM: Efficient stochastic transformer based world models for reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=WxnrX42rnS`.

Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024. URL `https://arxiv.org/abs/2405.12399`.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. Deepmind control suite, 2018. URL `https://arxiv.org/abs/1801.00690`.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data, 2021. URL `https://arxiv.org/abs/2111.00210`.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15)*, November 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015. doi: 10.1038/nature14236. URL `https://doi.org/10.1038/nature14236`.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning, 2015. URL `https://arxiv.org/abs/1507.04296`.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL `https://arxiv.org/abs/1606.01540`.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1, 2018. ISSN 2326-3865. doi: 10.1109/tkde.2018.2876857. URL `http://dx.doi.org/10.1109/TKDE.2018.2876857`.

Jing Yu Lim, Zarif Ikram, Samson Yu, Haozhe Ma, Tze-Yun Leong, and Dianbo Liu. Jedi: Latent end-to-end diffusion mitigates agent-human performance asymmetry in model-based reinforcement learning, 2025. URL `https://arxiv.org/abs/2505.19698`.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised state representation learning in atari, 2020. URL `https://arxiv.org/abs/1906.08226`.

A Technical Appendices and Supplementary Material

A.1 STORI Implementation

The STORI framework is built around a sophisticated wrapper-based architecture that introduces various types of uncertainty and partial observability into deterministic Atari environments with a granular control over the modifications.

A.1.1 Core Architecture and Wrapper System

The implementation uses a hierarchical wrapper system built on top of the Atari Learning Environment (ALE) (GPL-2.0 license). The main ‘StochasticEnv‘ class serves as the entry point, which applies different types of wrappers from ‘wrapper_registry‘ of specified environment. The system supports five distinct types, each introducing different forms of stochasticity. The system is highly configurable through a dictionary-based configuration system. Users can specify probabilities for different stochasticity effects, choose between different modes of operation, and configure temporal parameters for concept drift. The wrapper registry system allows for easy extension and customization of stochasticity types for new games or research requirements.

A.1.2 Stochasticity Wrappers

- Type 0: This type returns the RAM state of the game (a 1-D numpy array) with state labels as the observation. This implementation is an extension of Atari Annotated RAM Interface [Anand et al., 2020].
- Type 1: The ‘ActionDependentStochasticityWrapper‘ randomly replaces the agent’s intended action with a random action from the action space with a specified probability.
- Type 2: The ‘ActionIndependentRandomStochasticityWrapper‘ implements environment-specific random events that occur independently of the agent’s actions. These effects are applied probabilistically and create unpredictable environmental changes that the agent must adapt to. Read more about game-specific modifications in section A.1.3.
- Type 3: This introduces temporal concept drift where the environment dynamics change over time. The ‘ActionIndependentConceptDriftWrapper‘ supports both sudden and cyclic modes between 2 concepts. The concept 1 is the default environment (type 4) and concept 2 can be any other environment stochasticity types out of 1, 2 and 5. In sudden mode, the environment switches to concept 2 after a fixed number of steps. In cyclic mode, it alternates between the concept 1 and 2 every specified number of steps, creating a challenging environment where the agent must continuously adapt to changing dynamics.
- Types 4: This stochasticity type returns the default ALE environment without any modifications.
- Types 5: The ‘PartialObservationWrapper‘ introduces partial observability by modifying the agent’s observations. The system supports multiple observation modification techniques including cropping (removing portions of the screen), blackout (hiding specific regions), and RAM manipulation (temporarily modifying the game’s internal state to get modified observation).

In STORI, stochasticity types 1, 2, 3, and 5 are implemented as extensions of Type 4 environments. This is because screen-based observations serve as the default, well-studied ALE inputs for various reinforcement learning algorithms, providing a consistent foundation for comparing different types of stochasticity while also allowing for interpretable analysis of agent actions and behaviors.

A.1.3 Game-Specific Implementations

The implementation of type 1 and type 3 stochasticity wrappers is common to all games. The ‘crop‘ mode in type 5 stochasticity is also standard to all games, that includes the following modes:

- Mode 0: No crop
- Mode 1: Left - Crop the left half of the observation
- Mode 2: Right - Crop the right half of the observation

- Mode 3: Top - Crop the top half of the observation
- Mode 4: Bottom - Crop the bottom half of the observation
- Mode 5: Random circular mask - Randomly mask a circular region

Other game-specific modes for type 2 and type 5 stochasticity are listed in table 2.

A.2 Experiment Stochasticity Modes

For experiments, the different modes chosen for various stochasticity types in Breakout include:

- Type 1: Random action executed from action space instead of predicted action with a probability of 0.3.
- Type 2: If a block is hit, there is probability of 0.25 that the hit is not considered and the block is not destroyed thereby returning 0 reward and the ball bounces back.
- Type 3: Episode starts with default setting and after 300 steps into the episode, the dynamics suddenly change to *Type 5A*.
- Type 5A: The ball is only visible is a specific window between the blocks and the paddle and permanently hidden ($p = 1.0$) in rest of the space between them.
- Type 5B: Randomly hide left vertical half of the screen 75% ($p = 0.75$) of the episode.
- Type 5C: Only a random circular area of the screen is visible every frame ($p = 1.0$) similar to what someone will see when walking in a dark room with a torch.

Similarly, the different modes chosen for various stochasticity types in Boxing include:

- Type 1: Random action executed from action space instead of predicted action with a probability of 0.3.
- Type 2: Swaps the color of the enemy and player (character and score) with probability of 0.001 which results in 6-7 persistent swaps per episode (2 mins boxing round).
- Type 3: Episode starts with default setting and after 300 steps into the episode, the dynamics suddenly change to *Type 5C*.
- Type 5A: Permanently hide ($p = 1.0$) scores and game clock.
- Type 5B: Randomly hide right vertical half of the screen 75% ($p = 0.75$) of the episode.
- Type 5C: Randomly hide enemy character 70% ($p = 0.7$) of the episode.

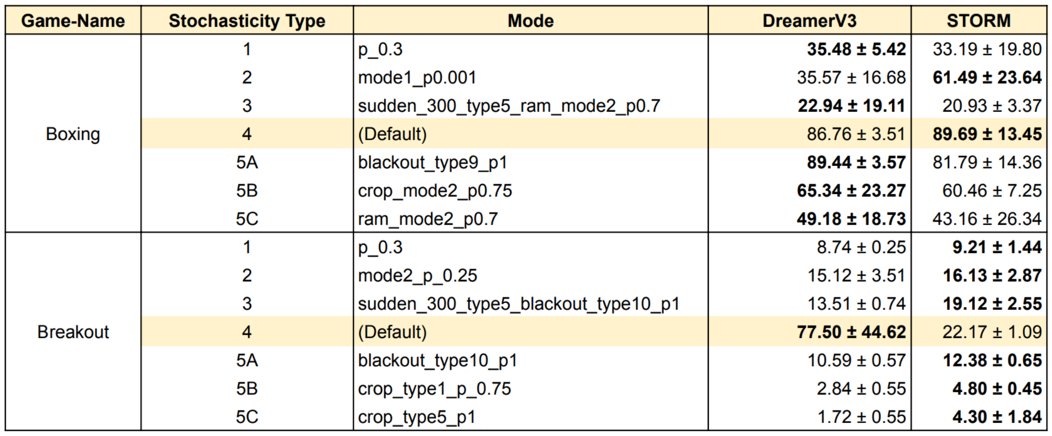
A.3 Algorithms Additional Details

- DreamerV3: The source implementation and default parameters for Atari100K config used from this code repository (MIT license): `https://github.com/NM512/dreamerv3-torch`
- STORM: The source implementation and default parameters (except eval num_episode was set to 100) used from this code repository: `https://github.com/weipu-zhang/STORM`

A.4 Full Result

Table 2: Environment modifications: Action-Independent Random Stochasticity, Partial Observation (Blackout and RAM Modification) for Boxing and Breakout

Game	Mode	Description
Boxing – Action-Independent Random Stochasticity (Type 2)		
	0	No stochasticity
	1	Colorflip - Swaps player and enemy colors (character and score)
	2	Hit cancel - Reverts score changes when hits occur
	3	Displace to corners - Randomly moves player and enemy positions
Boxing – Partial Observation: Blackout (Type 5)		
	0	No blackout
	1	All - Black out boxing ring, enemy score, player score, clock
	2	Left boxing ring
	3	Right boxing ring
	4	Full boxing ring
	5	Enemy score
	6	Player score
	7	Enemy score and player score
	8	Clock
	9	Enemy score, player score, and clock
Boxing – Partial Observation: RAM Modification (Type 5)		
	0	No modification
	1	Hide boxing ring
	2	Hide enemy
	3	Hide player
Breakout – Action-Independent Random Stochasticity (Type 2)		
	0	No stochasticity
	1	Block hit cancel
	2	Block hit cancel (reward reverted)
	3	Regenerate hit block
Breakout – Partial Observation: Blackout (Type 5)		
	0	No blackout
	1	All - Black out blocks, paddle, score, ball missing regions
	2	Blocks only
	3	Paddle only
	4	Score only
	5	Ball missing top
	6	Ball missing middle
	7	Ball missing bottom
	8	Blocks and paddle
	9	Blocks and score
	10	Ball missing top and bottom
	11	Ball missing all (top, middle, bottom)
Breakout – Partial Observation: RAM Modification (Type 5)		
	0	No modification
	1	Hide specific blocks
	2	Ball hidden - Hide the ball by setting its RAM values to 0



Game-Name	Stochasticity Type	Mode	DreamerV3	STORM
Boxing	1	p_0.3	**35.48 ± 5.42**	33.19 ± 19.80
	2	mode1_p0.001	35.57 ± 16.68	**61.49 ± 23.64**
	3	sudden_300_type5_ram_mode2_p0.7	**22.94 ± 19.11**	20.93 ± 3.37
	4	(Default)	86.76 ± 3.51	**89.69 ± 13.45**
	5A	blackout_type9_p1	**89.44 ± 3.57**	81.79 ± 14.36
	5B	crop_mode2_p0.75	**65.34 ± 23.27**	60.46 ± 7.25
	5C	ram_mode2_p0.7	**49.18 ± 18.73**	43.16 ± 26.34
Breakout	1	p_0.3	8.74 ± 0.25	**9.21 ± 1.44**
	2	mode2_p_0.25	15.12 ± 3.51	**16.13 ± 2.87**
	3	sudden_300_type5_blackout_type10_p1	13.51 ± 0.74	**19.12 ± 2.55**
	4	(Default)	**77.50 ± 44.62**	22.17 ± 1.09
	5A	blackout_type10_p1	10.59 ± 0.57	**12.38 ± 0.65**
	5B	crop_type1_p_0.75	2.84 ± 0.55	**4.80 ± 0.45**
	5C	crop_type5_p1	1.72 ± 0.55	**4.30 ± 1.84**

Figure 4: Full results for comparison of DreamerV3 and STORM performance across different stochasticity types, with Type 4 serving as the baseline (without any added stochasticity).

A.5 Compute Resources Used

The experiment runs were executed in several types of GPUs like A40, A100 and H100 depending on availability. The each node atleast had 32 vCPU and 50GB RAM. On GPUs with large memory, mulitple runs were executed.

For Breakout, DreamerV3 and STORM took around 24 hours and 12 hours respectively and for Boxing, DreamerV3 and STORM took around 16 hours and 9 hours respectively per run (training & evaluation) per seed when running on single GPU.