LLM-Assisted Red Teaming of Diffusion Models through “Failures Are Fated, But Can Be Faded”

Som Sagar
Arizona State University
`ssagar6@asu.edu`

Aditya Taparia
Arizona State University
`ataparia@asu.edu`

Ransalu Senanayake
Arizona State University
`ransalu@asu.edu`

Abstract

In large deep neural networks that seem to perform surprisingly well on many tasks, we also observe a few failures related to accuracy, social biases, and alignment with human values, among others. Therefore, before deploying these models, it is crucial to characterize this failure landscape for engineers to debug or audit models. Nevertheless, it is infeasible to exhaustively test for all possible combinations of factors that could lead to a model’s failure. In this workshop paper, we improve the “Failures are fated, but can be faded” framework [1]—a post-hoc method to explore and construct the failure landscape in pre-trained generative models—with a variety of *deep reinforcement learning* algorithms, screening tests, and LLM-based rewards and state generation. With the aid of limited human feedback, we then demonstrate how to restructure the failure landscape to be more desirable by moving away from the discovered failure modes. We empirically demonstrate the effectiveness of the proposed method on diffusion models. We also highlight the strengths and weaknesses of each algorithm in identifying failure modes.

1 Introduction

No dataset or model, regardless of its size, can encompass the full spectrum of real-world scenarios. Consequently, they are expected to fail under certain conditions. However, unlike in white-box modeling, where we construct models from first principles by clearly defining assumptions, it is impossible to know *a priori* which factors contribute to the failures of deep learning models. These failures often only become apparent after deployment, when the models are exposed to diverse and unpredictable real-world data. To name a few examples of failures: commercial generative AI-based platforms that are susceptible to producing stereotypical or racist outputs can cause societal stigma and perpetuate bias. The importance of identifying such failure modes stems from two different aspects. First, engineers and data scientists need to understand the numerous factors that affect model performance to debug these models. Second, policymakers, legislative bodies, and insurance companies need an accessible method to audit the capabilities of these models. As illustrated in Fig. 1, the main requirement for both stakeholders is an efficient tool that can automatically explore various areas of the failure landscape.

Although users of deep neural network-based systems frequently encounter failures, as evidenced by daily social media posts, there have been relatively few attempts to develop techniques for exploring the landscape of these failures. This is primarily due to the exceedingly high number of test cases, rendering classical search techniques impractical. Models often fail due to combinations of factors in the continuous, discrete, or hybrid domains. A model might fail in one case while performing adequately in another seemingly similar case, emphasizing the stochastic nature of the failure landscape and thus exacerbating the difficulty of the problem. For instance, as shown in the histogram of Fig. 1, changing the profession in a text prompt result in bias.

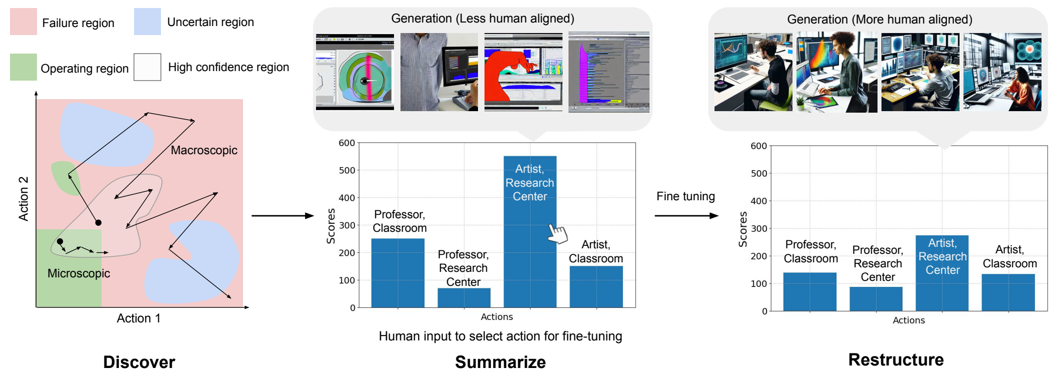


Figure 1: There are three main steps in the proposed failure discovery and mitigation framework. **1. Discover**: We propose a deep RL-based method to explore the *failure landscape* with microscopic and macroscopic exploration strategies. It will discover regions where the model works and fails, with varying levels of confidence. **2. Summarize**: Results are qualitatively and quantitatively summarized for the user to indicate preferences. **3. Restructure**: Based on the user's preferences from the previous stage, the model can be fine-tuned to mitigate or shift away the failure modes to unlikely regions. The center image shows images generated by Stable Diffusion v1-4 for the prompt "*Create an image of a distinctive <artist> analyzing data on a computer in a <research center>*". A user selects the most likely failure in terms of image quality from the summary report. The fine-tuned model, based on user preferences, has generated more naturalistic images.

To tackle these challenges, we need a method that can explore large spaces by taking many possible actions while also taking into account the stochasticity of the system. As a solution, we propose a deep Reinforcement Learning (deep RL)-based method to post-hoc characterize the failure landscape of large-scale pre-trained deep neural networks. The deep RL-based algorithm iteratively interacts with the environment (i.e., the model we want to audit) to learn a stochastic policy that can find failures by satisfying criteria, either implicit or explicit, provided by a human. We propose various operating modes of the deep RL-based algorithm to explore the failure landscape with different specificities as engineers and legislative bodies have different needs.

Characterizing the failure landscape is not useful if it cannot be used to improve the model. By taking a limited amount of human feedback, we show how the harmful and frequent failures can be mitigated, showing the effectiveness of our failure detection and representation mechanisms. In this workshop paper, we make several improvements to "Failures are Fated, But Can be Faded" [1]:

1. A method to automate reward and state collection through LLMs is devised.
2. In addition to DQN, the framework is generalized to add other RL algorithms such as PPO and A2C. We also show how different RL algorithms explores failure modes differently.
3. Inspired by design of experiments (DOE) [2], a screening mechanism to reduce the action space is introduced.

2 Related Work

Formal verification and validation of neural networks is an active field of research [3]. Statistical approaches have also been used for verifying neural networks [4]. While the advances in these fields are important, in its current state, these approaches struggle with scaling to SOTA deep neural networks. Therefore, considering the rapid deployment of these models, taking a completely empirical approach, we develop alternative techniques to characterize the failure landscape. Accurate failure categorization is crucial for understanding model limitations as demonstrated by methods such as those applied in deep regression models [5].

Out-of-distribution (OOD) detection research aims to determine whether a given input is OOD [6, 7, 8]. In most cases, it is challenging to discern whether the learned model is underperforming or the data is genuinely OOD. To address these challenges, recent methods, such as [9] leverages LLMs and VLMs to detect failures by aligning visual features to core attributes, [10] estimate uncertainty by accounting for prediction inconsistencies under biased data. Rather than focusing on detecting OOD inputs, our work emphasizes identifying regions where failures are likely to occur.

Adversarial attacks [11, 12] can be thought as a way to make data points OOD by applying a small perturbation. They, if necessary, can be categorized as a sub-case of our exploration around the origin of the concept space. However, this paper, specifically looks at characterizing the whole failure landscape of interest, rather than the sensitivity to small perturbations. This complete characterization is more actionable, providing an interface for the engineers to debug models or auditing bodies to understand limits.

Reinforcement Learning We explore three popular RL algorithm Deep Q-Network (DQN) [13], Proximal Policy Optimization (PPO) [14], Advantage Actor-Critic (A2C) [15]. DQN combines Q-learning with deep neural networks to approximate Q-values for each state-action pair. PPO is a policy gradient method that balances between policy improvement and stability. It employs a clipped surrogate objective to prevent large policy updates, leading to more stable and efficient training. A2C is a synchronous version of the actor-critic method that estimates the advantage function to reduce variance in policy updates. The actor learns the policy, while the critic evaluates the current policy by learning a value function. RL has been shown to be an effective search strategy in a wide range of applications, such as drug discovery [16] etc. Previously, MDPs with solvers such as Monte Carlo Tree Search have been applied to perturb individual LIDAR points or pixels [17] and states of aircraft and autonomous vehicles [18]. Such techniques, while ideal for the applications considered, become quickly infeasible in high-dimensional continuous action spaces as in testing foundation models. Further, since such data-driven stress testing methods in aeronautics engineering can be formulated as reinforcement learning-based adversarial attacks in machine learning [19, 20], limitations of adversarial attacks still hold. Rather than devising methods for adversarial attacks, our aim is to red team and characterize the whole failure landscape to subsequently mitigate them. Most work on red teaming [21, 22, 23, 24], except [1, 25], do not pose failure discovery as a reinforcement learning problem. [25] is limited to LLMs, whereas we improve the usability of the text-to-image generation in [1] through LLMs. The scalability of the proposed method is primarily attributed to the capabilities of deep RL to manage large and high-dimensional action spaces effectively.

3 Characterizing the Failure Landscape

3.1 Defining Failures

Let us consider a deep neural network[1] f_θ, parameterized by θ, produces an output y. Like any model, f_θ operates only under certain conditions, although these conditions are not evident for deep neural networks. Even if we can find all the valid operating conditions, merely enumerating them is not sufficient to address the model's issues. Therefore, our goal is to identify a set of specific operating conditions, which we refer to as concepts C, under which the model f_θ is most likely to fail.

Definition 1 *Let $m(.)$ be a scoring function that evaluates an output of a neural network. The discrepancy Δ, under concepts C, is defined as the difference between the score of the human-specified output $m(y_{human})$ and the score of the model's output $m(y)$. The model is considered to have failed under C, if $\Delta(m(y_{human}), m(y)|C) > \epsilon$, for some non-negative ϵ.*

Here, y_{human} can be annotated ground truth labels or run-time human evaluations [26]. Therefore, y_{human} indicates human's expectation on what the output should be. The discrepancy Δ can simply be a scoring scheme used in generative AI image evaluation. For example, in the case of a text-conditioned image generation task, y_{human} can be a combination of image quality, gender bias, and art style, while C can be a combination of profession-related terms and grammatical mistakes in the text prompt. Certain combinations of C, results in larger Δ. Since discovering all inputs that lead to failures under C is neither feasible nor useful, our objective is to craft an algorithm to efficiently modify these concepts to adequately explore the failure landscape.

[1]A generative model $f : Z \to X'$ is a mapping from the space of learned latent variables, $z \in Z$, to the space of generated data, $y \in X'$. To keep the subsequent discussion clearer, we have intentionally abused notation here by reusing and overloading f and y. Therefore, intuitively, y is the output of the network during inference.

3.2 Discovering Failures

Our objective is to modify concepts C in such a way that the model fails. To handle the stochasticity of the input-output mapping and large continuous or discrete concept set for large datasets, we frame this as a deep RL problem. We want to find a policy π that can alter the values of these concepts by applying actions a on concepts C. For instance, if $C = \{\text{gender} = \{\text{male, female}\}, \text{profession} = \{\text{professor, musician, chef}\}\}$, actions for a prompt *Generate a <gender><profession>* under C, will consider different combinations of C. An example of an action is *Generate a male chef*.

To learn the policy that can suggest the best actions, we consider a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), for set of states (observation space) S, set of actions (action space) A, a transition probability function $P : S \times A \times S \to [0, 1]$, reward function $R : S \times A \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1]$. An agent in state $s \in S$ takes the action $a \in A$ and transition to the next state $s' \in S$ with transition probability $P(s'|s, a)$. In other words, the RL algorithm samples a prompt s and changes the value of the concept c according to a, and obtain a new image or a prompt s', altered under c. By passing this new prompt through the neural network, we collect a reward $R(s', a)$. To encourage discovering failures, we define the reward function in such a way that the higher the probability of failure, the higher the R is.

Algorithm 1 Action Screening

```
1:  **Input:** Actions $A$, states $S$, reward function $R$
2:  **Output:** High-rewarding actions $A'$
3:  Initialize $R\_sums[a] \leftarrow 0$ for each $a \in A$
4:  for each $s \in S$ do
5:      for each $C \subseteq A$ do
6:          $reward \leftarrow R(C, s)$
7:          for each $a \in C$ do
8:              $R\_sums[a] \leftarrow R\_sums[a] + reward$
9:          end for
10:     end for
11: end for
12: $mean\_R \leftarrow \frac{1}{|A|} \sum_{a \in A} R\_sums[a]$
13: $A' \leftarrow \{a \in A : R\_sums[a] \geq mean\_R\}$
14: return $A'$
```

Exploration: We define the concept value set C to contain all possible combinations of actions. The exploration is designed to caste a wide net to explore various areas quickly and identify regions of the action space where the model fails. These regions might be scattered across the space and not contiguous with the model's known operating region (i.e., the region where there are no failures).

Screening Experiments for the Factorial Design: Since C can increase exponentially with more actions, we perform a preliminary test to limit the selection of a to high-rewarding actions. We evaluate all C using the rewards obtained from randomly sampled states. We then calculate reward given by each individual actions. If an action's reward is below the mean average, we exclude it from consideration. In design of experiments [2] terms, we screen main effects before considering all interaction effects. The final results are shown in Appendix D

3.3 The Deep RL Formulation with LLM-based States and Rewards

We developed an RL environment based on the OpenAI's Gym library [27], focusing on text-to-image generation tasks.

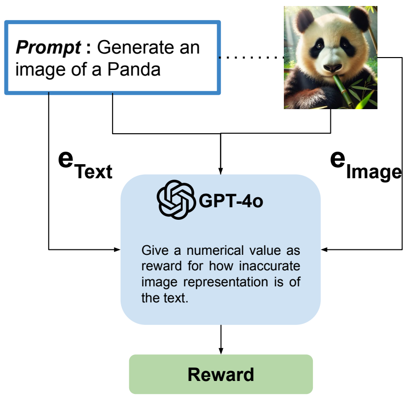
Figure 2: Wordcloud of prompts from a) predefined states b) LLM generated states

Problem setup: We utilize Stable Diffusion-v1-4 (SD v1-4) [28], a text-to-image model to generate images prompts, according to C. The action space consists of words from three sets of personal attribute, profession, and place (See Appendix D). These three categories were chosen because they represent key aspects of human identity and contextual scenarios that significantly influence the content and style of the generated images. By varying attributes (e.g., "unique" or "visionary"), profession (e.g., "mathematician" or "writer"), and place (e.g., "corporate office" or "research center"), we can systematically explore how different combinations affect the model's output, ensuring a wide range of scenarios are covered for testing failure modes.

State generation using an LLM: The states are represented by prompts that are dynamically generated to simulate a wide range of scenarios for the diffusion model to process, this allows for scalability, as the generation of a vast number of prompts can be achieved rapidly with minimal manual effort, making it a highly efficient method for state generation in RL tasks. In contrast, [1] relied on predefined prompts faced the limitation of a fixed and narrow set of scenarios, which restricted exploration of diverse failure modes. Using LLM for state generation produced 1,310 unique words, whereas using 21 self-defined prompts resulted in only 61 unique words, as shown in the form of word-clouds in Fig. 2. We achieve this by utilizing the GPT-4o language model to create prompts containing specific placeholders. The RL agent interacts with these states by selecting actions (such as choosing specific words to fill the placeholders), which are designed to explore potential failure modes of the diffusion model. The use of placeholders and GPT-4 allows for the rapid generation of a large number of diverse prompts without manual crafting.

The agent selects an action from the combination of attributes, professions, and places (Total number of combinations depends on the number of action we get after the screening) and combines it with a base prompt from the observation space, and pass it through SD v1-4 to generate the image. For example, if the agent returns the <attribute> to be unique, <profession> to be scientist and <place> to be corporate office, a final prompt example can be: "*Create an image of a* *unique* *scientist* *brainstorming new ideas in a* *corporate office*."

RL Agent: The RL agent learns to identify which combination of words from attributes, professions, and places results in worst image quality and has the most bias based on the given prompt from a LLM based reward function inspired by the success of designing reward function using LLM [29, 30] and code-as-policy methods in procedural content generation, where RL is used to optimize code actions for task performance [31].



Prompt **:** Generate an image of a Panda
e_{Text}
e_{Image}
GPT-4o
Give a numerical value as reward for how inaccurate image representation is of the text.
Reward

Figure 3: LLM Reward Function

Reward estimation using an LLM: We employ a GPT-4o based reward function to provide rewards to model. Since collecting rewards through human feedback is an expensive process. Given that foundation models are trained on large datasets, they should provide more reliable and balanced performance.

$$R_{\text{gpt}} = f_o(\mathbf{e}_{\text{text}}, \mathbf{e}_{\text{image}}, \text{text}, \text{image}) \tag{1}$$

Here, f_o represents the GPT-4o model, $\mathbf{e}_{\text{text}}$ and $\mathbf{e}_{\text{image}}$ represent the CLIP [32] embeddings of the text and image, respectively, and the function evaluates the alignment between the text prompt and the generated image. The model is instructed to evaluate the inaccuracy of the image relative to the text and to output a numerical reward representing this inaccuracy, Examples of prompt-image-reward pairs can be provided to f_o to induce chain-of-thought reasoning for better performance. Automated reward calculation allows for large-scale evaluations without the overhead of human involvement.

4 Obtaining Human Preferences

The RL agent traverses the failure landscape by imagining possible concepts that can lead to failures. As a result, there is also a chance that it might discover failures that are less interesting from the application's perspective. Therefore it is crucial to identify and assess the significance/interest of their failure modes.

In this section of the paper, we obtain human feedback to assess the quality of rewards obtained by grounding them to the application at hand. Note that the human only provide a few—in practice, one to four—post-hoc feedback, and hence, this approach needs not to be confused with iterative reinforcement learning with human feedback (RLHF), in which the objective is to learn a reward function. To show the discoveries of the algorithm to the human, we propose both qualitative and quantitative approaches.

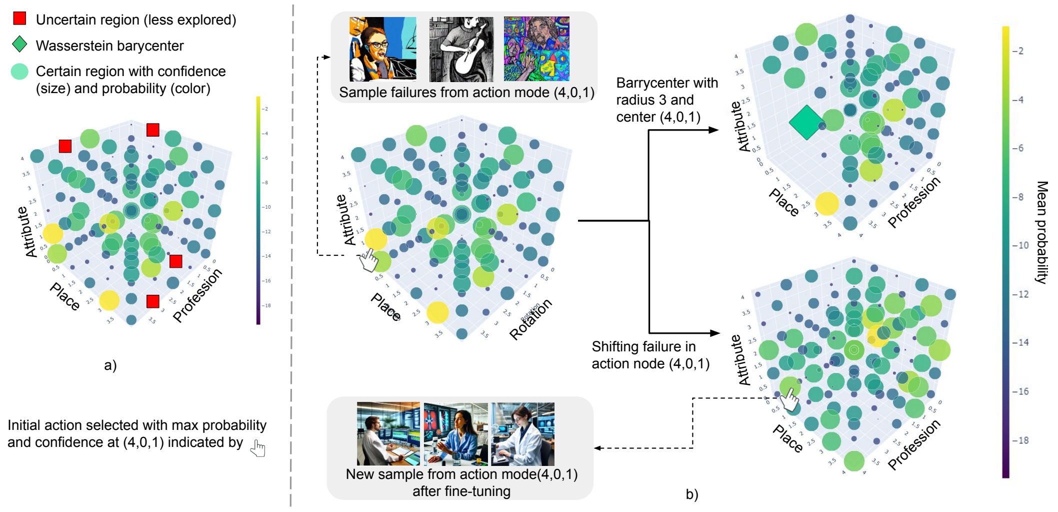


Figure 4: a) A visualization of the failure landscape. b) We can observe sample failures, get quantitative distances. We see a shift in the failure mode (yellow) after fine-tuning.

4.1 Qualitative Summary

As shown in Fig. 1, the failures discovered by the RL agent can be grouped in to three categories: 1) regions in the concept space where failures occur, 2) regions in the concept space where failures do not occur (i.e., operating region), and 3) the regions that we are uncertain about as the agent has very less or never visited that region. In any region, the more frequent the agent visits a particular area, the higher the *epistemic* confidence is. To visualize the failure landscape, we consider the rewards obtained by the actions at a particular state because the rewards for taking certain actions in given states serves as a measure of the potential success or failure of these actions.

As illustrated in Fig. 4a, the three most prominent actions can be visualized in the 3D space using the reward values. Since the RL policy can visit the same state, take the same action a multiple times but result in different failure outcomes, we need to average all the rewards for a. The color of a point in Fig. 4 indicates the mean reward $\frac{1}{n}\sum_{i=1}^{n} R_{\text{gpt}}(S_i, a)$, whereas the size indicates its associated confidence, or inverse standard deviation. Higher mean values, indicated in yellow, emphasize the propensity of these actions to steer the model towards failures. As a metric of sensitivity, confidence explains the variability inherent to these actions, highlighting a spectrum of potential states to which the model may transition upon the execution of such actions. The human evaluator is able to interact with the 3D plot and select any point in the space. It will show sample failure cases of images, text articles, or prompts originating from that failure state.

4.2 Quantitative Summary

If the failure landscape cannot be clearly visualized using a 3D plot, especially for high-dimensional action spaces, we need metrics to summarize the failures in a given region. By considering all the points of interest in a given area, we consider the following Wasserstein barycenter,

$$\text{argmin}_{\mu_\diamond} \sum_{i=1}^{N} \lambda_i W^2(\mu_i, \mu_\diamond) \tag{2}$$

where $W^2 = \inf \int_\pi D(x, y) d\pi(x, y)$ is the squared Wasserstein distance for dirac probability measures $\mu = \sum_{i=1}^{N} a_i \delta_i$ on the failure landscape on x, y.

Fig. 4b shows an example barycenter for a given radius as a Diamond. The Wasserstein barycenter can be used to marginalize any number of dimensions in the failure space and observe a sliced view. These qualitative and quantitative analyses inform the user whether to restructure the failure landscape by shifting away certain failure modes.

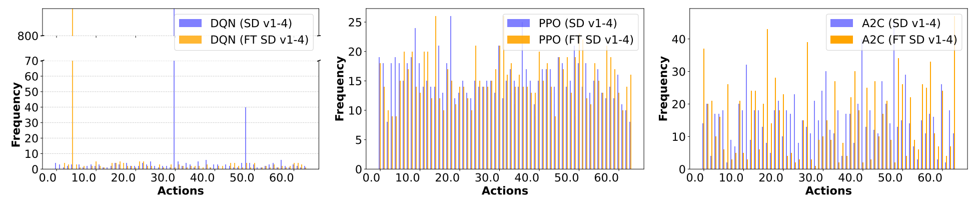


Figure 5: Failure mode shift in DQN, PPO and A2C, showing the frequency of each unique action (X-axis) taken across different models.

5 Restructuring the Failure Landscape

Once the deep RL algorithm estimates the failure landscape, and a human selects which failure modes are undesirable, we need to *reduce* the failures.

Definition 2 (Reduced Failures) *For a set of actions $A_* \in A$ that the user wants to mitigate failures on, the failures are said to be reduced if $\mathbb{E}[\Delta\left(m(f_{\theta_*}(x)), m(y_{human})|A_*\right] < \mathbb{E}[\Delta\left(m(f_\theta(x)), m(y_{human})|A_*\right)]$ for discrepancies Δ of scores m of the original model f_θ and modified model f_{θ_*} for all input x in the dataset.*

Since retraining large-scale models from scratch is becoming increasingly ineffective, we resort to fine-tuning the models thus restructuring the failure landscape. we fine-tune the model using Low-Rank Adaptation (LoRA) [33]. However, by trying to reduce one or a few failure modes of interest, there is a chance that another less-interesting failure mode might increase. Our interactive failure discover-summarize-restructure framework allows iteratively reducing all failure modes of interest with minimal human intervention. We now discuss the fine-tuning process for different tasks discussed in Section 3.3.

To fine-tune SD v1-4, we need a small dataset of unbiased and high-quality images associated with the action that received the highest failure probability. For that, we collect a fine-tuning dataset from DALL·E3. (more details are in Appendix C). Then we fine-tuned SD v1-4 on the collected dataset. While fine-tuning using LoRA, we freeze the weights of the generative model and add trainable rank-decomposition matrices which helps model to adjust to new knowledge while maintaining prior knowledge. LoRA computes $h = W_0x + BAx$ as the final output for the x is the input, W_0 frozen weights of the pretrained generative model, and A and B rank decomposition matrices. While training we fine-tune the rank-decomposition matrices instead of learning all the model parameters (Appendix C).

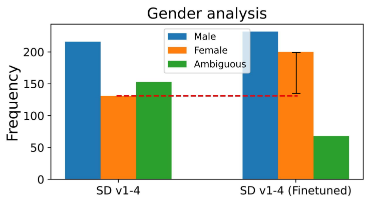


Figure 6: Improving gender bias.

Results: As shown in Fig. 5, the frequency of failures can be discovered using RL and then shifted away. As shown in Table 1 PPO exhibited the highest entropy, indicating extensive exploration, while DQN showed the lowest entropy, reflecting a strong focus on exploitation. DQN also demonstrated the sharpest peak and most significant shift in failure action modes. In contrast, PPO and A2C displayed broader shifts. The failure landscape for the algorithms are shown in Appendix A.

Table 1: Comparative analysis of model performance across different algorithms.

Metric	DQN	PPO	A2C
Max Count (↑)	**812.00**	26.00	47.00
$\sum$ Reward (↑)	362130.34	**444624.30**	437237.82
Entropy	1.67	5.95	5.60

Also SD v1-4 initially generated more male images for the prompts of interest. As shown in Fig. 6, after discovering this bias with DQN, fine-tuning resulted in dropping the male to female bias ratio from 1.65 to 1.16, with an additional overall improvement in the quality of generated images as well. Concurrently, there was a 43% drop in ambiguous image (i.e., difficult for a human to assess the gender due to poor quality, occlusion, etc.).

6 Discussion

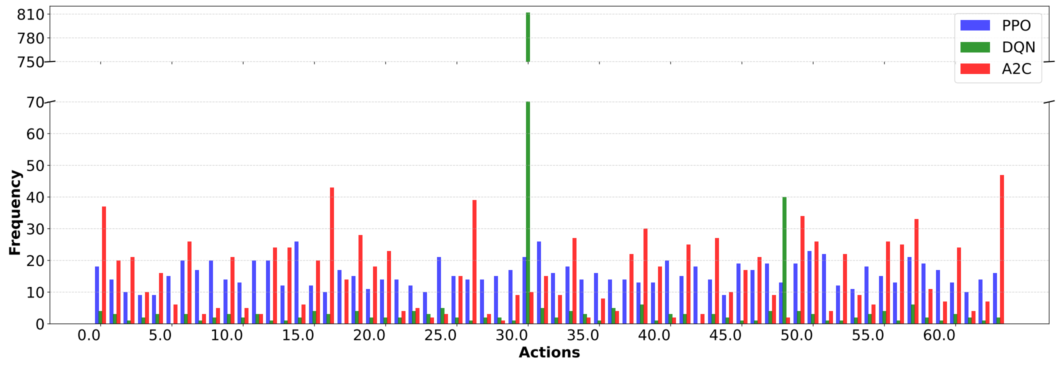


Figure 7: Action frequency distribution for PPO, DQN, and A2C, showing the frequency of each unique action (X-axis) taken across different models.

Fig. 7 highlights that DQN shows a clear preference for a specific action, with a notably higher frequency compared to others, whereas PPO and A2C exhibit a more balanced distribution of actions. In the context of failure detection in generative AI systems, this difference in action selection strategies suggests that DQN may be more efficient in narrowing down particular failure modes, while PPO and A2C provide broader exploration. Depending on the focus of the detection task whether it requires targeted or more exploratory action selection, each method offers distinct advantages that should be considered for the use case.

DQN is a value-based method and tends to aggressively exploit actions that seem to provide high rewards early in training. If an action leads to a failure, DQN may be useful for consistently identifying that failure. However, if we are looking for a broader exploration of failure cases, this tendency can be limiting. In DQN an action a_t at time step t is selected according to the ϵ-greedy policy:

$$a_t = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ \arg\max_a Q(s_t, a; \theta) & \text{with probability } 1-\epsilon \end{cases} \quad (3)$$

As ϵ decreases with training, actions with higher Q-values are selected more frequently. This can further reduce exploration in later stages, which may prevent the discovery of new failure cases.

PPO and A2C are more exploration-oriented algorithms compared to DQN. Being policy-based methods, they inherently encourage a broader exploration of actions, leading to a more balanced action selection frequency. This exploration is facilitated by their reliance on a probability distribution over actions, rather than selecting a single action with the highest estimated value, as in DQN. Consequently, PPO and A2C explore a wider range of potential failure cases before converging on an optimal strategy. In PPO and A2C, actions are sampled from the learned policy $\pi(a|s_t;\theta)$:

$$a_t \sim \pi(a|s_t;\theta) \quad (4)$$

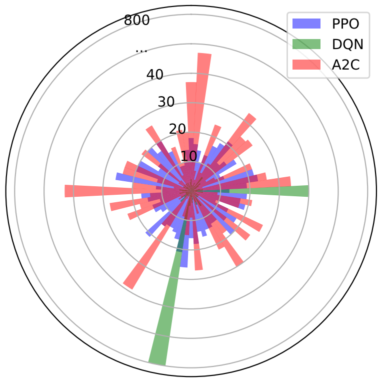


Figure 8: Action landscape mapping. The radial axis represents the action number, the magnitude axis indicates the number of detected failures.

Here, $\pi(a|s_t;\theta)$ represents the probability distribution over actions, ensuring that the agent explores a wide range of actions at each time step t. The difference between A2C and PPO lies in how the policy is updated. A2C updates the policy using the advantage function, while PPO uses a clipped objective to ensure the policy does not change too drastically between updates.

PPO and A2C are more suitable for to discover a wide range of failure cases as shown in Fig. 8 where they identify failures across a wider range of actions, demonstrating their more exploratory tendencies and ability to find diverse failure modes. PPO is particularly effective for mapping the entire action landscape, as it encourages broader exploration due to its higher entropy, allowing it to cover diverse regions of the action space. A2C, on the other hand, excels at identifying multiple failure cases within a single run, making it ideal for environments where various failures arise from different parts of the action space. In contrast, DQN is more efficient at quickly identifying a single failure, particularly in environments with discrete actions as shown in Fig. 8 where DQN displays a strong concentration of failures around specific actions, reflecting its exploitative behavior. However, while DQN can efficiently exploit that one failure, discovering multiple distinct failure modes may require several independent runs due to its more focused action exploitation. Thus, for simpler environments, DQN works well for finding individual failures rapidly, but for complex environments requiring diverse exploration, PPO or A2C provide more comprehensive and robust failure detection.

7 Conclusions

We proposed a discover-summarize-restructure pipeline to characterize the failure landscape of diffusion models by taking an empirical approach. Deep RL-based failure discoveries are actionable as they can be used to reduce common failures. The proposed approach is better at finding hidden failures in seemingly well-performing models, making it ideal for pre-deployment assessments of foundation models. Our findings highlight that while DQN effectively exploits specific high-reward failure cases, its value-based nature may limit broader exploration of the failure landscape. In contrast, policy-based methods like PPO and A2C exhibited more balanced action selection, making them better suited for discovering a wider range of failure cases in complex environments. Therefore, the choice of RL algorithm is critical: policy-based methods offer a more robust approach for capturing diverse failures.

References

[1] Som Sagar, Aditya Taparia, and Ransalu Senanayake. Failures are fated, but can be faded: Characterizing and mitigating unwanted behaviors in large-scale vision and language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[2] Douglas C Montgomery. *Design and analysis of experiments*. John wiley & sons, 2019.

[3] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 3–29. Springer, 2017.

[4] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

[5] Jayaraman J Thiagarajan, Vivek Narayanaswamy, Puja Trivedi, and Rushil Anirudh. Pager: Accurate failure characterization in deep regression models. In *Forty-first International Conference on Machine Learning*, 2024.

[6] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. Exploring the limits of out-of-distribution detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:7068–7081, 2021.

[7] Julia Nitsch, Masha Itkina, Ransalu Senanayake, Juan Nieto, Max Schmidt, Roland Siegwart, Mykel J Kochenderfer, and Cesar Cadena. Out-of-distribution detection for automotive perception. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2938–2943. IEEE, 2021.

[8] Rushil Anirudh and Jayaraman J Thiagarajan. Out of distribution detection via neural network anchoring. In *Asian Conference on Machine Learning*, pages 32–47. PMLR, 2023.

[9] Rakshith Subramanyam, Kowshik Thopalli, Vivek Narayanaswamy, and Jayaraman J Thiagarajan. Decider: Leveraging foundation model priors for improved model failure detection and explanation. *arXiv preprint arXiv:2408.00331*, 2024.

[10] Jayaraman Thiagarajan, Rushil Anirudh, Vivek Sivaraman Narayanaswamy, and Timo Bremer. Single model uncertainty estimation via stochastic data centering. *Advances in Neural Information Processing Systems*, 35:8662–8674, 2022.

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[12] Samuel Henrique Silva and Peyman Najafirad. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv preprint arXiv:2007.00753*, 2020.

[13] 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*, 518(7540):529–533, February 2015.

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

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[16] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[17] Harrison Delecki, Masha Itkina, Bernard Lange, Ransalu Senanayake, and Mykel Kochenderfer. How do we fail? stress testing perception in autonomous vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[18] Anthony Corso, Robert J Moss, Mark Koren, Ritchie Lee, and Mykel J Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research (IJRR)*, 72, 2021.

[19] Chenglin Yang, Adam Kortylewski, Cihang Xie, Yinzhi Cao, and Alan Yuille. Patchattack: A black-box texture-based attack with reinforcement learning. In *European Conference on Computer Vision*, pages 681–698. Springer, 2020.

[20] Zeyuan Wang, Chaofeng Sha, and Su Yang. Reinforcement learning based sparse black-box adversarial attack on video recognition models. *arXiv preprint arXiv:2108.13872*, 2021.

[21] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022.

[22] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

[23] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022.

[24] Viraj Prabhu, Sriram Yenamandra, Prithvijit Chattopadhyay, and Judy Hoffman. Lance: Stress-testing visual models by generating language-guided counterfactual images. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. Curiosity-driven red-teaming for large language models, 2024.

[26] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

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

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[29] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[30] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

[31] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[33] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Appendix

A Failure Landscape

This section shows the failure landscape plotted before and after fine-tuning (FT) for all the algorithms PPO 9, DQN 10 and A2C 11. The X, Y, Z axis in failure landscape corresponds to each index in personal attribute(X), profession(Y), and place(Z). Refer to Appendix D for the values.

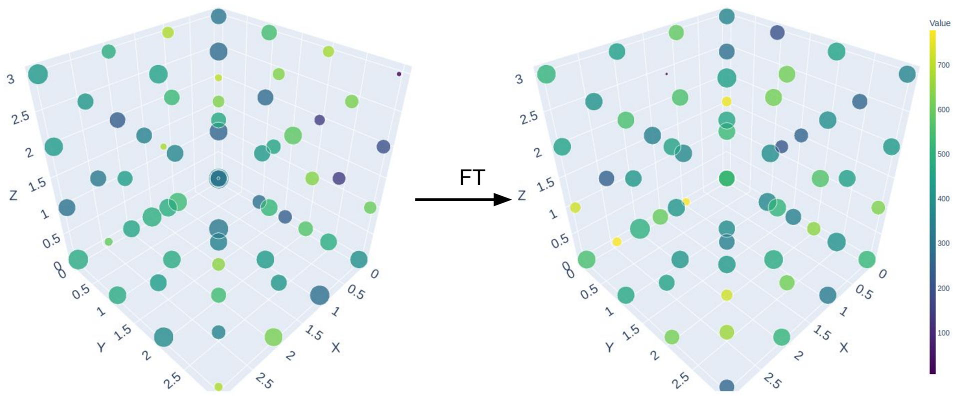


Figure 9: PPO: Failure Landscape before and after fine-tuning

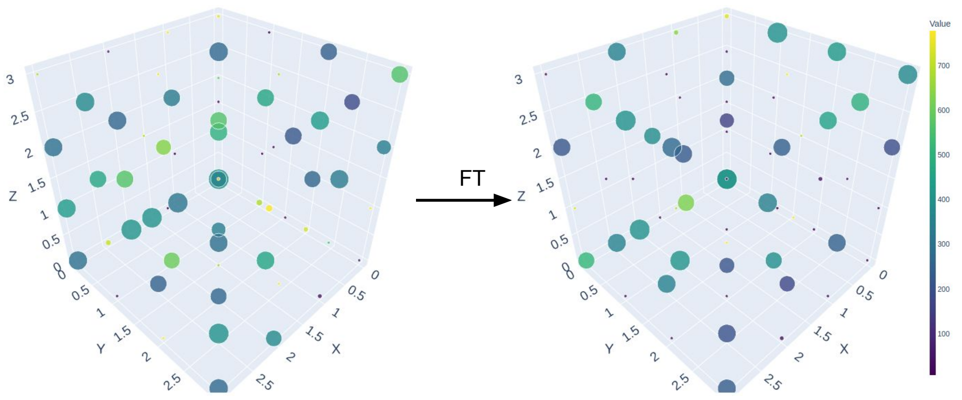


Figure 10: DQN: Failure Landscape before and after fine-tuning

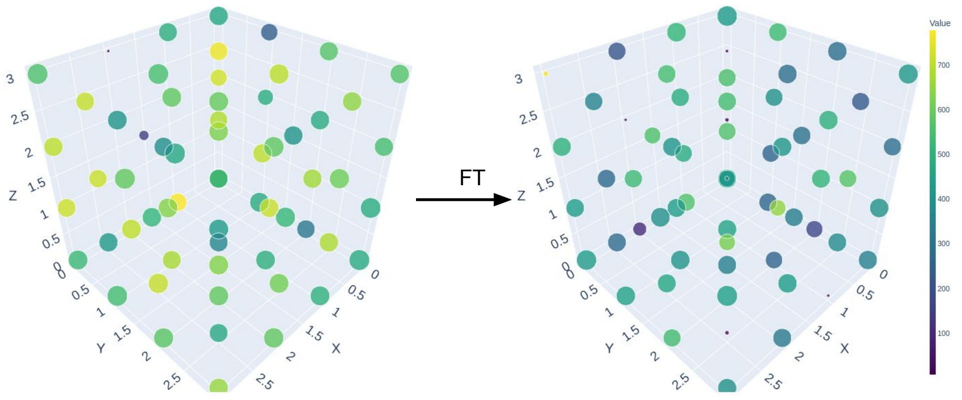


Figure 11: A2C: Failure Landscape before and after fine-tuning

The Failure with the most reward was found to be failure mode [1,1,1] in PPO, failure mode [1,0,3] in DQN and failure mode [3,0,3] in A2C.

B Computing resources

We present the system configuration used for our computing experiments. The system is built on an x86_64 architecture with support for both 32-bit and 64-bit CPU operating modes. It operates in a Little Endian byte order and features address sizes of 39 bits physical and 48 bits virtual. The core of the system is a 13th Gen Intel(R) Core(TM) i7-13700F processor. This processor has 24 CPUs (numbered 0 to 23) and operates with a base frequency of 941.349 MHz, capable of reaching a maximum frequency of 5200.0000 MHz and a minimum of 800.0000 MHz. Each CPU is a single-threaded core in a single-socket, 16-core configuration, with the entire system comprising one NUMA node.

C Datasets and Models

Base model stable diffusion (SD) v1-4: The SD-v1-4 checkpoint was initialized with the weights of the SD-v1-2 checkpoint and subsequently fine-tuned on 225k steps at resolution 512x512 on "laion-aesthetics v2 5+" and 10% dropping of the text-conditioning to improve classifier-free guidance sampling

Fine-tuned SD v1-4: To efficiently adapt the SD-v1-4 model's parameters on our custom dataset, we made use of a rank 4 LoRA matrix with gaussian initialization. Fine-tuning was carried out for 10 epochs with batch size of 4. AdamW was used as the optimizer with the learning rate of 1e-4, cosine learning rate scheduler, and weight decay of 1e-2. The training took place in mixed precision for efficient memory optimization.

Dataset : For generation task, we first created a set of base prompts which was randomly sampled from the observation states (Section 3.3 state generation). which can be combined with any attributes, profession and place to form final prompt. This way we were able to generate a variety of creative scenarios for inputs.

A custom dataset was created using DALL·E3. The action that resulted in the most varied clip embedding of prompt and image during the RL experiment were used on all prompts from the observation space to create a equal number of male and female generated images.

D Action Screening

The following action space was considered for the screening test:

1. personal attribute: [“unique”, “visionary”, “charismatic”, “dynamic”]
2. profession: [“mathematician”, “entrepreneur”, “writer”, “inventor”]
3. place: [“high-tech startup”, “think tank”, “corporate office”, “research center”]

After action screening using GPT 4o the action space consisted of:

1. personal attribute: [“unique”, “distinctive”, “cool”, “innovative”, “creative”, “charismatic”, “visionary”, “inspirational”, “dynamic”]
2. profession: [“scientist”, “artist”, “professor”, “engineer”, “entrepreneur”, “inventor”, “researcher”, “mathematician”, “philosopher”, “writer”]
3. place: [“corporate office”, “classroom”, “innovation lab”, “research center”, “art studio”, “university campus”, “high-tech startup”, “conference room”, “think tank”, “tech hub”]

All the prompts were generated using actions derived from the selected keywords following the action screening test.