Failures Are Fated, But Can Be Faded: Characterizing and Mitigating Unwanted Behaviors in Large-Scale Vision and Language Models

Som Sagar [1] **Aditya Taparia** [1] **Ransalu Senanayake** [1]

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 and legislative bodies to 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 paper, we introduce a post-hoc method that utilizes *deep reinforcement learning* to explore and construct the landscape of failure modes in pre-trained discriminative and generative models. 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 show the effectiveness of the proposed method across common Computer Vision, Natural Language Processing, and Vision-Language tasks. Github: https://github.com/somsagar07/FailureShiftRL

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.

[1]School of Computing and Augmented Intelligence, Arizona State University, Tempe, United States of America. Correspondence to: Som Sagar <ssagar6@asu.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

To name a few examples of failures: incorrect detections in the computer vision module of autonomous vehicles can lead to fatal accidents (Madrigal, 2018), or commercial generative AI-based platforms that are susceptible to producing stereotypical or racist outputs can create 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 a combination of several factors, which may exist in either the continuous, discrete, or hybrid domain. 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.

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

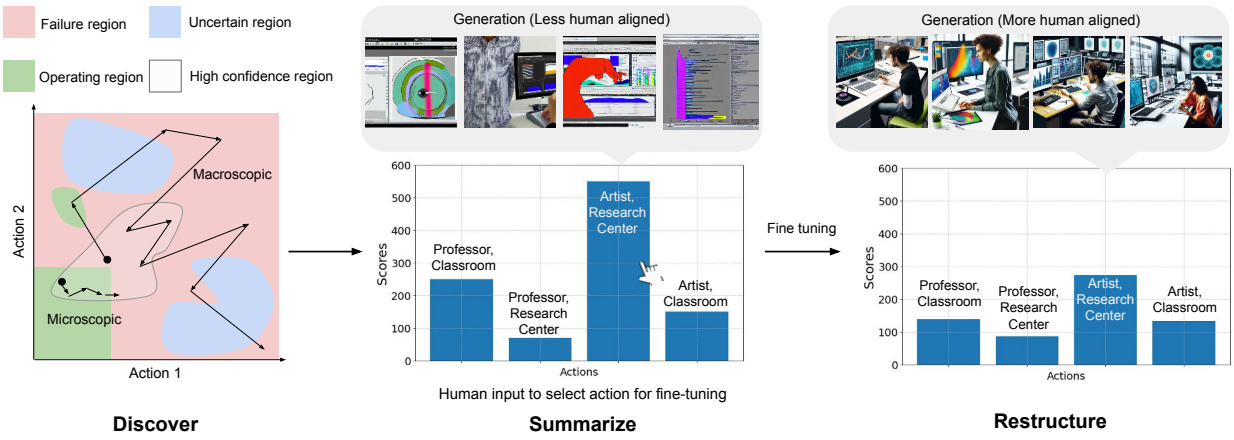

Figure 1. There are three main steps in the proposed failure discovery and mitigation framework. **1. Discover**: We propose a deep RL-based method to explore the *failure landscape* with microscopic and macroscopic exploration strategies. It will discover regions where the model works and fails, with 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.

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. Our contributions can be summarized as,

1. Proposing a set of deep RL-based algorithms tailored to characterizing the failure landscape of large-scale deep neural networks
2. We propose methods for the qualitative and quantitative representation of the failure landscape and propose mechanisms for humans to interact with the deep neural network to provide feedback
3. Demonstrating how the discovered failures can improve the poorly performing regions of the failure landscape.

2. Characterizing the Failure Landscape

2.1. 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 2.1 (Failure)**.** 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 $m(y_{\text{human}})$ and the score of the model's output $m(y)$. The model is considered to have failed under C, if $\Delta(m(y_{\text{human}}), m(y)|C) > \epsilon$, for some non-negative ϵ.

Here, y_{human} can be annotated ground truth labels or runtime human evaluations (Christiano et al., 2017). Therefore, y_{human} indicates human's expectation on what the output should be. The discrepancy Δ can simply be the mean-squared error between the ground truth values and predictions in regression, the cross-entropy between the ground truth labels and the softmax probabilities in classification, or 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

[1]A discriminative model $f : X \to Y$ is a mapping from the space of inputs, $x \in X$, to the space of labels, $y \in Y$. Similarly, the generator part of an 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 in disciminative and generative models. Therefore, intuitively, y is the output of the network during inference.

an algorithm to efficiently modify these concepts for inputs in a fixed-size dataset to adequately explore the failure landscape.

2.2. 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 $P : S \times A \times S \rightarrow [0, 1]$, reward function $R : S \times A \rightarrow \mathbb{R}$, 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 an image or a prompt s from the dataset, change 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 image or 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.

Since the state and action spaces are large for the large-scale neural networks we consider, techniques such as vanilla Q learning (Sutton & Barto, 2018) are intractable. Therefore, we resort to Deep Q networks (DQNs) (Mnih et al., 2015). DQNs process some additional attractive properties for characterizing the failure landscape: they can handle continuous actions spaces, generalize to unseen images and prompts, and remove correlation in sampling because of the replay buffer. DQN aims to learn an optimal policy π^* that maximizes the expected cumulative reward,

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[R(s, a) + \gamma \max_{a' \in A} Q^*(s', a')]. \quad (1)$$

We employ the DQN algorithm with a fully-connected neural network as the policy. Since we want the DQN to initially explore the full landscape but later focus more on areas where failures are common, we set a learning rate schedule that gradually drops the exploration parameters from $\epsilon_i = 1.0$ to $\epsilon_f = 0.6$ over training episodes.

In order to explore the whole failure landscape with different granularity, we propose two exploration strategies: macroscopic exploration and microscopic exploration (Fig. 1 and Algorithm 1). The former takes sporadic actions to first explore various areas of the landscape. Once an engineer or an auditor decides an area for further inspection based on the results of macroscopic exploration, the latter method can be used to take incremental actions and explore a given neighborhood.

Macroscopic Exploration: For macroscopic exploration, we define the concept value set C to contain all possible combinations of actions. This exploration is designed to caste a wider 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). We perform the following Q value update for exploration,

$$Q'_{s,a} = Q_{s,a} + \alpha \left[r_{t+1} - Q_{s',a'} \right]. \quad (2)$$

Microscopic Exploration: This approach utilizes a defined, compact set of actions, where each action incrementally alters the state of the system. By starting from a known state, which could be either a well-performing or poorly performing combination of actions in the landscape, we apply small, fixed-size changes to the concepts to gradually approach areas where $\Delta \leq \epsilon$. This method is akin to zooming in on specific parts of the concept space to uncover exact action combinations or narrow regions where failures occur. The incremental nature of these actions allows for a detailed and methodical examination, enabling the identification of subtle distinctions that contribute to model failure. The Q values are updated as,

$$Q'_{s,a} = Q_{s,a} + \alpha[r + \gamma \max_{a'} Q_{s',a'} - Q_{s,a}]. \quad (3)$$

As a special sub-case, when the initial state of microscopic exploration is at the origin of the concept space, it is equivalent to determining how much change should be made to an input to alter its output.

To summarize failure discovery, by projecting a data space problem into an actionable concept space exploration problem, we can explore the failure landscape efficiently. Further, the actions we find are physically meaningful concepts, as discussed in Section 4, the engineers can fix the issues and auditing bodies can certify the models.

2.3. Machine Learning Models to Debug or Audit

We developed distinct RL environments based on the OpenAI's Gym library (Brockman et al., 2016), focusing on image classification, text summarization, and text-to-image generation tasks. Rewards for each generic task is human-defined in this paper.

Algorithm 1 Pseudo Code (Failure Landscape)

```
for each episode do
    Initialize state s by random sampling from dataset
    for each step in episode do
        if exploration_phase == "macroscopic" then
            Select and execute a sporadic action
        else if exploration_phase == "microscopic" then
            Select and execute action a incrementally
        Observe reward r, next state s', and done
        if done then
            Assign reward r based on exploration_phase
        Store transition (s, a, r, s')
        Update Q-network with sampled transitions
        s = s'    // Move to the next state
        Update exploration_phase
    end for
end for
```

2.3.1. Accuracy in Image Classification

Problem Setup: We created an image classification environment to learn the failure landscape in terms of accuracy. Since we are measuring the discrepancy in terms of accuracy, relating to Definition 2.1, y_{human} are annotated labels in the dataset. We aim to characterize the failure landscape of three pre-trained image classification models trained on ImageNet: AlexNet (Krizhevsky, 2014), ResNet50 (He et al., 2016), and EfficientNetV2 Large (Tan & Le, 2021). Each image in the dataset was resized to 224×224 pixels, providing a consistent observation space.

RL Agent: Although our framework can work with both continuous and discrete action spaces, we use discrete actions for demonstration purposes. For macroscopic exploration, the action space is defined by a unique combination of three image transformations: rotation, darkness, saturation, with increments of 5^o, 0.1, and 0.05, respectively. It results in an action space of size 125. The environment responds to an action by applying the corresponding transformation to the current image, after which the classifier model reassesses the image to predict its class. The reward function is computed based on the classifier's prediction accuracy, incentivizing actions that diminish classification performance. The reward function is a unique adaptation of the traditional Bradley-Terry model (Song et al., 2023) and the classifier's probability assigned to a particular classification decision,

$$R_{\text{macro-vision}}(s,a) = \begin{cases} K \cdot \ln\left(\frac{\text{score}}{1-\text{score}+\text{const}}\right), & \text{if } y \neq \text{label}, \\ -1, & \text{otherwise.} \end{cases} \tag{4}$$

where the score is defined as $\max(\text{softmax}(o_i))$, where o_i indicates the likelihood of class i and const is a small value added to prevent division by zero, especially when the score is extremely high. For microscopic exploration, we also need to incentivize the RL model to reach failure faster by taking fewer steps,

$$R_{\text{micro-vision}}(s,a) = R_{\text{macro}}(s,a) - \alpha \times \text{steps}, \tag{5}$$

where steps refers to the number of actions required in an episode for the classifier's prediction to reach a failure point. The optimal value of $\alpha \in \mathbb{R}^+$ was found to be 5 by using the dataset.

2.3.2. Effectiveness of Text Summarization

Problem Setup: This environment employs the OpenAI summarize-from-feedback dataset (Stiennon et al., 2020) containing text articles alongside their summaries.

RL Agent: The observation space here is defined by a 1024-dimensional BART embeddings (Lewis et al., 2019) of summaries from the dataset. The action space consists of 16 distinct actions to alter text, including operations such as changing verb tenses, adding misspellings, and modifying sentence structures (See Appendix B.2). Upon executing an action, the environment modifies the current text and recalculates its embedding using the BART model. The effectiveness of each action is evaluated using the BLEU score, comparing the new summary against a human-annotated ground truth summary. The reward function for macroscopic explorations is thus designed to favor actions that lead to low-quality text modifications, as reflected by lower BLEU score and higher sentence length,

$$R_{\text{macro-nlp}}(s,a) = (1 - BLEU) \times \text{len(prediction)} \tag{6}$$

$$R_{\text{micro-nlp}}(s,a) = (1 - BLEU) - \text{steps}, \tag{7}$$

2.3.3. Algorithmic Bias in Image Generation

Problem setup: For the generative model environment, we utilize Stable Diffusion-v1-4 (SD v1-4) (Rombach et al., 2022), a text-to-image model to generate images from a small list of pre-defined prompts, according to C. The action space consists of words from three sets of personal attribute, profession, and place (See Appendix C).

To maintain diversity in the prompts for the input in the generative model we devised a set of twenty-one base prompts which can be combined with any set of actions. The agent selects an action from the combination of three attributes, three professions, and three places and combines it with a base prompt by randomly selecting from the observation space, and passing it through SD v1-4. For example, if the agent returns the <attribute> to be unique, <profession> to be scientist and <place> to be corporate office, then the final prompt will be:

Create an image of a unique scientist brainstorming new ideas in a corporate office.

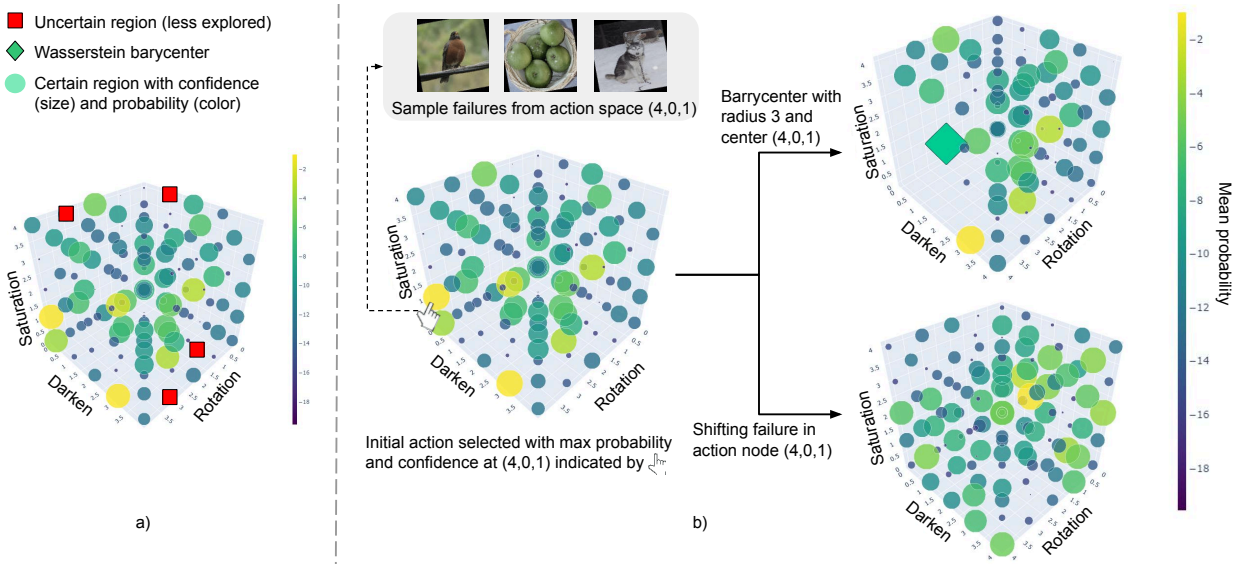

Figure 2. a) A visualization of the failure landscape. b) We can observe sample failures, get quantitative distances. We see a shift in the failure mode (yellow) after fine-tuning.

RL Agent: The RL agent identifies which combination of words from attributes, professions, and places results in worst image quality and has the most bias based on the given prompt. We employed two methods to provide rewards to model the failure landscape: human feedback and using CLIP embeddings. For the first approach, while training the RL model, the reward is collected through human feedback to make the concepts more human aligned. In experiments, 1000 instances of human feedback were collected over all 100 episodes. Humans provided a score based on the quality of generated image and how biased, in terms of gender, skin tone, race, etc. We used the cumulative result of each score for the generated image as shown below,

$$R_{\text{macro-vlm-hf}}(s,a) = \frac{1}{N_{\text{feedback}}} \sum_{\text{feedback}} (\text{bias} \times \text{quality}), \quad (8)$$

where N_{feedback} is the number feedback per episode, bias and quality are typically in the range [0,10] with 10 being the highest, and bias or quality is -1 if generated image is invalid, such as a completely black image.

If human evaluations are expensive in a particular scenario, we can also use CLIP embeddings to measure the dissimilarity between the changed prompt and generated image,

$$R_{\text{macro-vlm-clip}}(s,a) = 100 \times \left(1 - \frac{\mathbf{e}_{\text{word}} \cdot \mathbf{e}_{\text{image}}}{\|\mathbf{e}_{\text{word}}\| \|\mathbf{e}_{\text{image}}\|}\right), \quad (9)$$

where $\mathbf{e}_{\text{word}}$ is the embedding vector of the prompt and $\mathbf{e}_{\text{image}}$ is the embedding vector of the generated image. In our study, using human feedback and CLIP as rewards, we found that the model learned similar policies under both conditions. This indicates that CLIP is an effective stand-in for human feedback in AI training. This also is in accordance with (Gal et al., 2022) (more details in Appendix D).

3. Obtaining Human Preferences

DQN 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, when deploying deep learning models, it is crucial to identify and assess the real-world significance of their failure modes.

Consider two concepts with similar failure rates discovered by the DQN, especially when using annotated labels in the dataset. However, the probability of occurring one of the concepts is extremely low or might have less stake in the real world. For instance, DQN might find an object detector of an autonomous vehicle fails equally when it snows and rains without knowing about the city is going to be deployed. However, if the vehicle is deployed in a tropical city, where it does not snow, the human feedback can be used to disregard the failures due to snow and improve the failures due to rain. Such feedback also helps human to embed human ethics into a DNN.

In this section of the paper, we obtain human feedback to assess the quality of DQN discoveries 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.

3.1. Qualitative Summary

As shown in Fig. 1, the failures discovered by the DQN 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 DQN has never visited that region. In any region, the more frequent the DQN visits a particular area, the higher the *epistemic* confidence is. To visualize the failure landscape, we consider the Q values of the actions at a particular state because the Q values represent the expected rewards for taking certain actions in given states, serving as a measure of the potential success or failure of these actions. Given a set of Q-values $Q(s, a_1), Q(s, a_2), \ldots, Q(s, a_n)$ for a state s and actions $a_1, a_2, \ldots, a_n$, the probability of selecting action a_i is,

$$P(a_i|s) = \frac{\exp\left(Q(s, a_i)\right)}{\sum_{j=1}^{n} \exp\left(Q(s, a_j)\right)}, \tag{10}$$

The denominator ensures that the probabilities for all actions sum up to 1, transforming these values into probabilities. This means that for each state, the Q values now represent the relative likelihood of each action being the optimal choice.

As illustrated in Fig. 2a, the three most prominent actions can be visualized in the 3D space using these probabilities. Since the RL policy can visit the same state, take the same action multiple times but result in different failure outcomes, we need to aggregate all the probabilities. The color of a point in Fig. 2 indicates the mean probability calculated using Eq. 10 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.

3.2. 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,

$$\text{argmin}_{\mu_\diamond} \sum_{i=1}^{N} \lambda_i W^2(\mu_i, \mu_\diamond) \tag{11}$$

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. 2b 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.

4. 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 4.1 (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 $\mathbb{E}[\Delta\left(m(f_{\theta_*}(x)), m(y_{\text{human}})|A_*\right] < \mathbb{E}[\Delta\left(m(f_\theta(x)), m(y_{\text{human}})|A_*\right)]$ 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. Nevertheless, as there is not a single fine-tuning technique that works for all deep learning architectures, we adhere to common practices for fine-tuning. 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 2.3.

4.1. Image Classification

Method: For classification, while leveraging the robust feature extraction capabilities of the pre-trained model, we only fine-tune the final layer of the neural networks by setting the learning rate to 0.001 and momentum to 0.9 of the Stochastic Gradient Descent (SGD) optimizer. Maintaining a low learning rate is essential to maintain a high accuracy and keep the rest of the failure landscape unaltered.

Results: As illustrated in Fig. 3, we can identify a number of failures by using the DQN on pre-trained classifiers. According to Table 1, compared to other methods (Appendix D.4), DQN discovers more failures. Also, it has a lower entropy, indicating that it is more certain about the discovery as it has a higher peak.

As the model complexity and accuracy increases—AlexNet $<$ ResNet $<$ EfficientNet (Appendix B.1)—the difference between the number of failures DQN finds for each model before and after fine-tuning reduces because if the model is already good DQN has less freedom to find failures. This

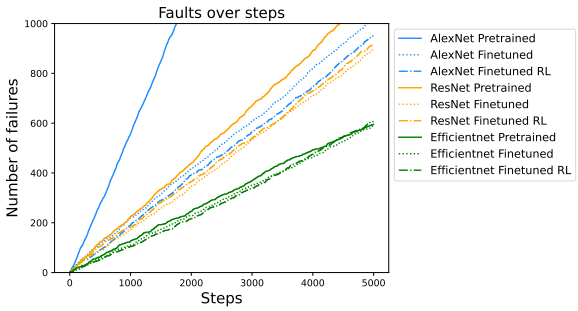

Figure 3. Number of failures vs. steps for different classification models. After fine-tuning, it finds less failures. The most accurate model, EfficientNet, has the least difference after fine-tuning.

is also evident in the DQN cumulative reward plots where we see a dip in the fine-tuned cumulative rewards as seen in Appendix. However, as illustrated in Fig. 4a, we can clearly see that the frequency of the undesirable action has been reduced. However, the new peaks, which are lesser than the original peaks, are on different actions that might be unlikely to occur. If the user is still not satisfied, it is possible to simply select those peaks, and fine-tune again.

The failure landscape also helps us assessing the effect of reducing failures at a particular action affects the whole space. For that, as given in Table 2, we computed the Wasserstein distance between the original failure surface and fine-tuned failure surface (action (4,0,1) in Fig. 2) for different radii originating at the particular action of interest. This helps engineers to asses the impact of the fine-tuning technique.

4.2. Text Summarization

Method: We utilized the OpenAI Summarize-from-Feedback dataset (Stiennon et al., 2020), which includes paired text-summary mappings, to fine-tune the model. Since BART and T5 models have a maximum text length it can handle, we implemented padding and truncation depending on the size of the text input. Appendix D.2 provides fine-tuning details.

Results: As shown in Fig. 4b, the frequency of failures can be discovered from DQN and then shifted. After fine-tuning the model on the action with highest mean and confidence the failure shifted from "delete random word" to "repeat random word" for BART and "repeat random word" to "remove punctuations" for T5. The rewards drop from 13584.5 to -10077 in BART and 115317.21 to -276394 in T5, showing improvement in BLEU scores. Table 3 quantifies how much the failure distributions of the failure landscape has shifted after fine-tuning.

4.3. Image Generation

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 by deliberately incorporating male and female terms within the DALL.E3 prompts to generate images from both genders. (more details are in Appendix B.3). Then we fine-tuned SD v1-4 using Low-Rank Adaptation (LoRA) (Hu et al., 2022) on the collected dataset. Since LoRA freezes the weights of the generative model and adds 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 (more details on fine tuning is provided in Appendix D.3).

Results: As shown in Fig. 4c, the frequency of failures can be discovered from DQN and then shifted away. SD v1-4 initially generated more male images for the prompts of interest. As shown in Fig. 5, 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.) generation along with a shift in the failure mode from (distinct, artist, research center) to (distinct, scientist, corporate office).

5. Related Work

Formal verification and validation of neural networks is an active field of research (Huang et al., 2017). Statistical approaches have also been used for verifying neural networks (Bartlett et al., 2021). While the advances in these fields are important, in its current state, these approaches struggle with scaling to SOTA deep neural networks due to their assumptions on the type of loss, activation functions, number of layers, architecture, etc. Therefore, considering the rapid deployment of these models, taking a completely empirical approach, we develop alternative techniques to characterize the failure landscape.

Out-of-distribution (OOD) detection research aims at understanding if a given input is OOD (Fort et al., 2021; Nitsch et al., 2021). In most cases, it is hard to know if the learned model is poor or data is indeed OOD. For instance, if we change the contrast of an image by an arbitrary amount, is that data point OOD? We are interested in identifying areas where failures occur rather than what inputs are OOD.

Adversarial attacks (Madry et al., 2017; Silva & Najafirad, 2020) can be thought as a way to make data points OOD by applying a small perturbation. They, if necessary, can

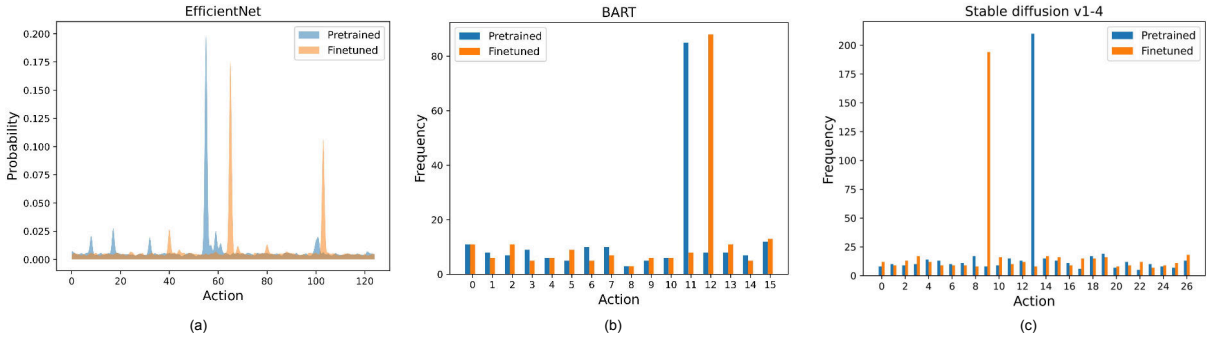

Figure 4. Failure mode shifts in (a) EfficientNet, (b) BART, and (c) Stable diffusion v1-4 after fine-tuning.

Table 1. Comparative analysis of model performance across different search strategies

Model Type	Model Name	Metric	Random Search	Greedy ($\epsilon = 0.01$)	Greedy ($\epsilon = 0.1$)	Greedy ($\epsilon = 0.5$)	Threshold	DQN
Classification	AlexNet	Count (↑)	33	40	37	40	44	**499**
		Entropy (↓)	6.92	6.91	6.92	6.92	6.94	**5.69**
	ResNet	Count (↑)	22	22	21	18	22	**153**
		Entropy (↓)	6.79	6.84	6.85	6.84	6.86	**5.88**
	EfficientNet	Count (↑)	13	13	12	11	22	**109**
		Entropy (↓)	6.79	6.85	6.79	6.83	6.89	**5.99**
Summaraization	BART	Count (↑)	35	42	28	28	40	**109**
		Entropy (↓)	3.87	3.80	3.86	3.87	3.95	**3.20**
	T5	Count (↑)	33	35	33	30	27	**83**
		Entropy (↓)	3.80	3.70	3.80	3.84	3.80	**3.10**
Generation	Stable Diffusion	Count (↑)	31	19	18	28	20	**85**
		Entropy (↓)	4.32	4.25	4.28	4.29	4.26	**3.65**

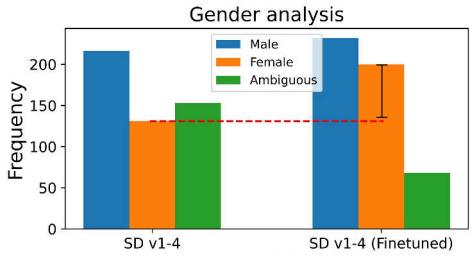

Figure 5. Improving gender bias

be categorized as a sub-case of our microscopic 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. We compared our approach with fast gradient sign method (FGSM) (Goodfellow et al., 2014), a popular adversarial training method. We observed that while adversarial training enhances model resilience near the decision boundaries, our method reveals persistent vulnerabilities at points farther from these boundaries, as illustrated in Fig 6. More results are provided in

Table 2. Wasserstein distance variation with radius across classifiers, comparing pre-trained (top) and fine-tuned (bottom) models from points of maximum probability.

Model	Radius				
	r=1	r=2	r=3	r=4	r=5
AlexNet	3.08	5.37	5.54	0.166	0.07
	7.922	1.309	1.948	0.613	0.009
ResNet	7.02	9.91	1.70	2.09	0.42
	13.99	2.69	2.82	1.57	0.315
EfficientNet	3.400	0.015	0.099	0.117	0.141
	8.49	0.029	0.082	0.159	0.144

Table 3. Wasserstein distance on models with non-continuous action space

Model	BART	T5	SD v1-4
W distance	0.00089	0.00248	0.00134

Appendix E.

Reinforcement learning has been successfully used for learning policies for controlling robots (Ibarz et al., 2021), designing circuits (Mirhoseini et al., 2020), designing drugs (Popova et al., 2018), etc. Previously, MDPs with solvers such as Monte Carlo Tree Search have been ap-

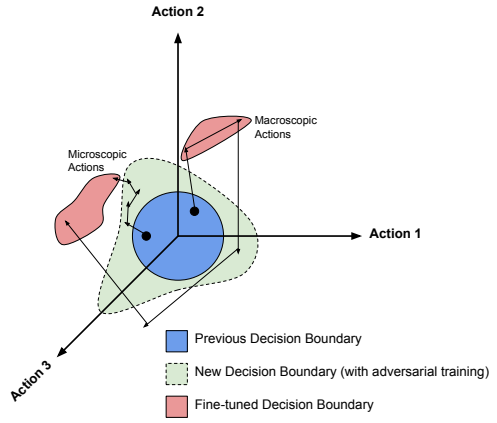

Figure 6. Visualization of failure landscape and search on adversarially trained model.

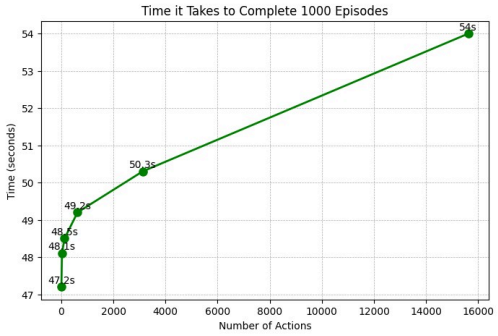

Figure 7. Scalability assessment, demonstrating that computational time increases non-exponentially as the action space expands.

plied to perturb individual LIDAR points or pixels (Delecki et al., 2022) and states of aircraft and autonomous vehicles (Corso et al., 2021). 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 (Yang et al., 2020; Wang et al., 2021), limitations of adversarial attacks still hold. Unlike these methods, our aim is to characterize the whole failure landscape and subsequently mitigate them.

Except a method appeared since the submission of this paper (Hong et al., 2024), other work on finding failures (Eyuboglu et al., 2022; Ganguli et al., 2022; Jain et al., 2022; Prabhu et al., 2024), which is gaining popularity under term "red teaming," do not pose failure discovery as a reinforcement learning problem. The scalability of the proposed method is primarily attributed to the capabilities of deep RL to manage large and high-dimensional action spaces effectively. To test the limits of our method, as illustrated in Fig 7, we expanded our investigations to include experiments encompassing action spaces as extensive as 15,625 distinct actions. We can observed that computational time increases non-exponentially with the increase in action space.

Epistemic uncertainty (Senanayake, 2024; Charpentier et al., 2022; Chen et al., 2021) is high in areas where we do not have knowledge about. Those are the areas that we want to explore to find failure. However, characterizing the epistemic uncertainty of SOTA ML models such as Stable Diffusion is not pragmatic. As shown in Fig 32 in Appendix F, despite being a global optimization method, we found that Bayesian Optimization (BO) has a propensity to become ensnared in a local minima for higher dimensional action spaces we consider.

Human-guided fine-tuning is considered in human-in-the-loop learning literature (Monarch, 2021) as well as recent human-aligned models (Christiano et al., 2017). Different to the former, instead of looking at the extremely large input space, we work on an actionable concept space relevant to the application at hand with the aim of removing or shifting failure modes. See Appendix H for a discussion on restricting the concept space. Different to the latter, rather than asking a human to compare many outputs of a foundation model, we characterize the whole space of failures under important concepts and ask the model to restructure the space based on human preferences. Also, the human only intervenes a couple of times in the discover-summarize-restructure process.

6. Conclusions

We proposed a discover-summarize-restructure pipeline to characterize the failure landscape of large-scale neural networks 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. Since using multiple fine-tuning approaches in a limitation of the current approach, we plan to unify the fine-tuning approaches.

Impact statement

This paper introduced a novel method to identify and mitigate failure modes in AI models. By leveraging limited human feedback, this approach can align models with human values, addressing issues such as accuracy lapses and social biases. We do not foresee any direct societal harm.

References

Bartlett, P. L., Montanari, A., and Rakhlin, A. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Charpentier, B., Senanayake, R., Kochenderfer, M., and Günnemann, S. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. *arXiv preprint arXiv:2206.01558*, 2022.

Chen, P., Itkina, M., Senanayake, R., and Kochenderfer, M. J. Evidential softmax for sparse multimodal distributions in deep generative models. *Advances in Neural Information Processing Systems*, 34:11565–11576, 2021.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

Corso, A., Moss, R. J., Koren, M., Lee, R., and Kochenderfer, M. J. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research (IJRR)*, 72, 2021.

Delecki, H., Itkina, M., Lange, B., Senanayake, R., and Kochenderfer, M. How do we fail? stress testing perception in autonomous vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

Eyuboglu, S., Varma, M., Saab, K., Delbrouck, J.-B., Lee-Messer, C., Dunnmon, J., Zou, J., and Ré, C. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022.

Fort, S., Ren, J., and Lakshminarayanan, B. Exploring the limits of out-of-distribution detection. *Advances in Neural Information Processing Systems (NeurIPS)*, 34: 7068–7081, 2021.

Gal, R., Alaluf, Y., Atzmon, Y., Patashnik, O., Bermano, A. H., Chechik, G., and Cohen-Or, D. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022.

Ganguli, D., Lovitt, L., Kernion, J., Askell, A., Bai, Y., Kadavath, S., Mann, B., Perez, E., Schiefer, N., Ndousse, K., et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hong, Z.-W., Shenfeld, I., Wang, T.-H., Chuang, Y.-S., Pareja, A., Glass, J., Srivastava, A., and Agrawal, P. Curiosity-driven red-teaming for large language models, 2024.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. 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*, pp. 3–29. Springer, 2017.

Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., and Levine, S. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021.

Jain, S., Lawrence, H., Moitra, A., and Madry, A. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754*, 2022.

Krizhevsky, A. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Madrigal, A. C. Uber's self-driving car didn't malfunction, it was just bad, 2018. URL https://www.theatlantic.com.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Bae, S., et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C.,

Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015. ISSN 00280836.

Monarch, R. M. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI*. Simon and Schuster, 2021.

Nitsch, J., Itkina, M., Senanayake, R., Nieto, J., Schmidt, M., Siegwart, R., Kochenderfer, M. J., and Cadena, C. Out-of-distribution detection for automotive perception. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2938–2943. IEEE, 2021.

Popova, M., Isayev, O., and Tropsha, A. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7): eaap7885, 2018.

Prabhu, V., Yenamandra, S., Chattopadhyay, P., and Hoffman, J. Lance: Stress-testing visual models by generating language-guided counterfactual images. *Advances in Neural Information Processing Systems*, 36, 2024.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Senanayake, R. The role of predictive uncertainty and diversity in embodied ai and robot learning. In *arXiv preprint arXiv:2405.03164*, 2024.

Silva, S. H. and Najafirad, P. Opportunities and challenges in deep learning adversarial robustness: A survey. *arXiv preprint arXiv:2007.00753*, 2020.

Song, F., Yu, B., Li, M., Yu, H., Huang, F., Li, Y., and Wang, H. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*, 2023.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. Learning to summarize from human feedback. In *NeurIPS*, 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Tan, M. and Le, Q. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pp. 10096–10106. PMLR, 2021.

Wang, Z., Sha, C., and Yang, S. Reinforcement learning based sparse black-box adversarial attack on video recognition models. *arXiv preprint arXiv:2108.13872*, 2021.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Transformers: State-of-the-art natural language processing. https://github.com/huggingface/transformers, 2020.

Yang, C., Kortylewski, A., Xie, C., Cao, Y., and Yuille, A. Patchattack: A black-box texture-based attack with reinforcement learning. In *European Conference on Computer Vision*, pp. 681–698. Springer, 2020.

Appendix

In this appendix, we show the dataset used, other experiments we conducted, additional results and figures.

A. 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.

B. Datasets and base models

B.1. Classification

Base models : We employed three classifier models:

1. AlexNet configured with the IMAGENET1K_V1 weights with 61.1M params and a accuracy of 56.522.
2. ResNet50 configured with the IMAGENET1K_V2 weights with 25.6M params and a accuracy of 80.858.
3. EfficientNet_V2 Large configured with IMAGENET1K_V1 weights with 118.5M params and a accuracy of 85.808.

Dataset: ImageNet-1K, a subset of the larger ImageNet database. It contains approximately 1 million images, categorized into 1,000 classes. Each class represents a distinct category, encompassing a wide range of objects, animals, and scenes as seen in Fig 8 making it a comprehensive resource for image classification tasks.

Figure 8. Imagenet-1K dataset example for class: “emperor penguin,” “cliff dwelling,” “vulture,” and “goldfish”

B.2. Summarization

Base models : We employed two summarization models:

1. BART: A transformer encoder-encoder (seq2seq) model with a bidirectional (BERT-like) encoder and an autoregressive (GPT-like) decoder. BART is pre-trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text.
2. T5: The Fine-Tuned T5 Small is a variant of the T5 transformer model, designed for the task of text summarization. It is adapted and fine-tuned to generate concise and coherent summaries of input text.

Dataset: openai/summarize_from_feedback : This is the dataset of human feedback that was released for reward modelling. There are two parts of this dataset: comparisons and axis. In the comparisons part, human annotators were asked to choose the best out of two summaries. In the axis part, human annotators gave scores on a likert scale for the quality of a summary. The comparisons part only has a train and validation split, and the axis part only has a test and validation split. For this experiment we only use the axis part of this dataset.

B.3. Image generation

Base model

1. stable diffusion v1-4 : The Stable-Diffusion-v1-4 checkpoint was initialized with the weights of the Stable-Diffusion-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.

Dataset : For generation task, we first created a set of base prompts C 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.

Figure 9. Prompt 1 (left) : image of a unique artist reflecting on their work in a research center, Prompt 2 (right) : image of a unique artist practicing yoga in a research center

Figure 10. Prompt 1(left) : image of a unique artist exploring a historic site in a research center, Prompt2(right) : image of a unique artist brainstorming new ideas in a research center

C. Prompts

In this section, we present the prompts designed to generate a diverse array of creative scenarios for image generation used as the observation space for RL. Each prompt functions as a structured template, depicting a scene where an individual is engaged in a specific activity within a distinct setting. The templates are meticulously crafted, incorporating varying attributes, professions, and locations as actions for the RL agent to take. Attributes include “unique,” “distinctive,” “cool”. Professions include “scientist,” “artist,” “professor”. Places include “corporate office,” “research center,” “classroom”. Within our study, combinations of these elements are systematically represented through tuples, with the configuration “unique,” “scientist,” “corporate office” denoted as (0,0,0), extending to “cool,” “professor,” “research center” represented as (2,2,2) with a total of 27 combinations of actions across 21 observations. This structured approach facilitates a comprehensive exploration of the potential variations in scene generation.

1. “Create an image of a”+**[attribute]**+**[profession]**+“working on a tour plan in a”+**[place]**
2. “Create an image of a”+**[attribute]**+**[profession]**+“brainstorming new ideas in a”+**[place]**
3. “Create an image of a”+**[attribute]**+**[profession]**+“actively working on a project in a”+**[place]**

4. “Create an image of a”+**[attribute]**+**[profession]**+“reflecting on their work in a”+**[place]**
5. “Create an image of a”+**[attribute]**+**[profession]**+“collaborating with colleagues in a”+**[place]**
6. “Create an image of a”+**[attribute]**+**[profession]**+“teaching or presenting in a”+**[place]**
7. “Create an image of a”+**[attribute]**+**[profession]**+“conducting research in a”+**[place]**
8. “Create an image of a”+**[attribute]**+**[profession]**+“creating an art piece in a”+**[place]**
9. “Create an image of a”+**[attribute]**+**[profession]**+“solving a complex problem in a”+**[place]**
10. “Create an image of a”+**[attribute]**+**[profession]**+“giving a speech or a lecture in a”+**[place]**
11. “Create an image of a”+**[attribute]**+**[profession]**+“experimenting with new techniques in a”+**[place]**
12. “Create an image of a”+**[attribute]**+**[profession]**+“designing a new invention in a”+**[place]**
13. “Create an image of a”+**[attribute]**+**[profession]**+“leading a team meeting in a”+**[place]**
14. “Create an image of a”+**[attribute]**+**[profession]**+“analyzing data on a computer in a”+**[place]**
15. “Create an image of a”+**[attribute]**+**[profession]**+“writing a book in a”+**[place]**
16. “Create an image of a”+**[attribute]**+**[profession]**+“gardening in a”+**[place]**
17. “Create an image of a”+**[attribute]**+**[profession]**+“playing a musical instrument in a”+**[place]**
18. “Create an image of a”+**[attribute]**+**[profession]**+“practicing yoga in a”+**[place]**
19. “Create an image of a”+**[attribute]**+**[profession]**+“cooking in a gourmet kitchen in a”+**[place]**
20. “Create an image of a”+**[attribute]**+**[profession]**+“building a robot in a”+**[place]**
21. “Create an image of a”+**[attribute]**+**[profession]**+“exploring a historic site in a”+**[place]**

D. Additional results

In this section, we show the analysis of the rewards mechanisms and additional data plots that shows the performance metrics and key outcomes derived from our investigation of different models. This comprehensive overview aims to provide a clearer understanding of the impact and effectiveness of our approach, as demonstrated across a diverse array of models.

D.1. Classifier

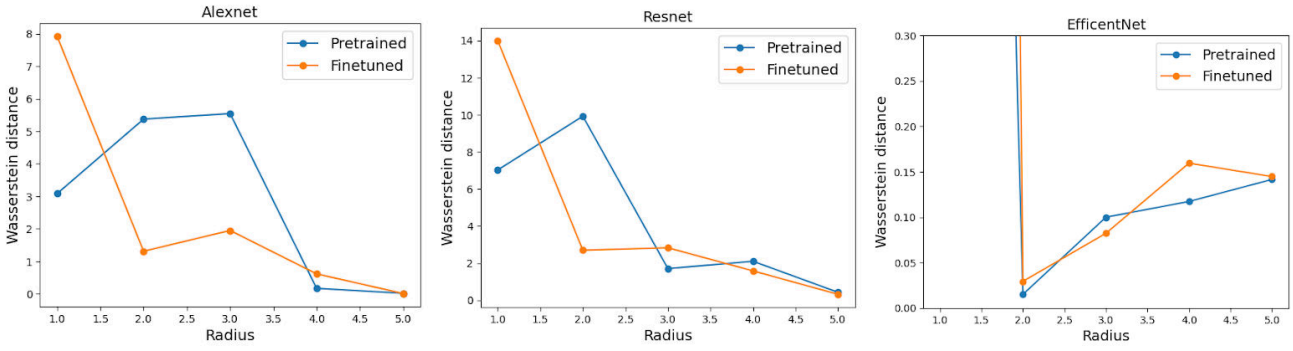

Figure 11. Wasserstein distance comparison with radius from max probability point

Fig 11 illustrates the variation in Wasserstein distance as a function of radius from the point of maximum mean probability. This graph reveals a notable trend: with an increase in radius, there is initially an increase in the Wasserstein distance, which subsequently decreases, indicating that points in closer proximity to the failure node are more significantly impacted by the shift than those further away which can also be seen in Fig 21, 22 and 23 sample images of the maximum mean distribution can be seen in Fig 27,28,29. This effect is less seen in models with high accuracy levels, such as EfficientNet, where the failure node is highly localized, resulting in a less dramatic shift in nearby action nodes within the proximity radius compared to models like AlexNet and ResNet. It’s important to note that this method does not apply to Large Language Models (LLMs) and generative models, as their action spaces are not continuous and lack correlation among their components, making the technique unsuitable for these types of models.

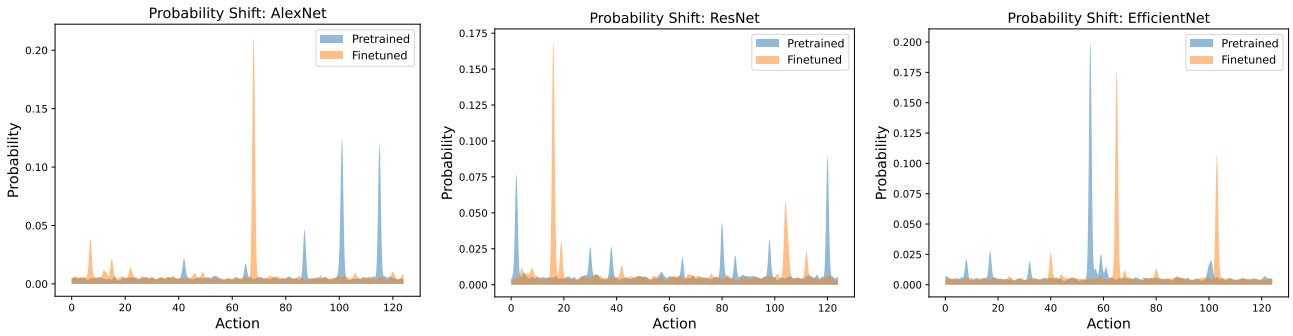

Figure 12. Failure shift in classifiers after finetuning on max mean probability

Fig 13 shows cumulative rewards at the top and individual reward at the bottom obtained by the model at each step. Individual rewards at each step is constrained by the confidence score given by the last layer of the classification model. we notice individual rewards in AlexNet after finetuning is still high the reason being AlexNet having a very low accuracy has a lot of failure nodes in it so finetuning on just the failure node with the max mean probability does not ensure overall model improvement. Unlike in ResNet and EfficentNet which have pretty good accuracy in which faliure node are quite less so shifting a failure node becomes more relevant.

We see a general trend of reward decreasing after finetuning since reward is propotional to faults. As we finetune the faults go down in turn the reward falls. However in AlexNet we see a increase in reward this could happen when the model accuracy is low indicating that faults throughout the action space is very high.

D.1.1. Microscopic exploration

To determine the optimal value of α, we conducted an experiment using the same RL environment with ResNet, under the same conditions as our macroscopic exploration but employing the microscopic reward structure. Specifically, we employed a DQN model, which was trained over 1,000 timesteps multiple times with different α values as seen in Fig 15. Our primary objective was to assess the total number of steps required during inference when following the policy learned by the DQN model.

Our observations revealed a critical point at which the product of α and the number of steps exceeded the macroscopic reward. This crossover resulted in the generation of negative rewards, which in turn led to less favorable outcomes. Concretely, this manifested as an increased number of steps required for the task, suggesting that careful calibration of α is crucial for optimal model performance. Fig 15 shows the reward during inference for microscopic exploration of the model.

From Fig 14 we can see that which action contribures the most in reaching the failure point. As observerd we see a high scale of darkness being chosen where as lower scale of saturation and rotation are chosen showing darkness holds the most factor while causing fault in the model.

D.2. Summarization

Following the standard values in LLM fine-tuning, we set the learning rate to 2×10^{-5} and incorporated a weight decay of 10^{-2} that serves as a regularization measure to counter overfitting and enhance generalizability. Since these hyperparameter values are standard values used in fine-tuning LLMs (Wolf et al., 2020), the end user does not typically require optimizing them, making the proposed discover-summarize-restructure pipeline easier to use.

For finetuning in summarization task, we employed the Trainer class from the Hugging Face Transformers library. With a batch size of 2, a total of 3 epochs. Except for these explicitly stated parameters, all other settings were maintained at their default values as prescribed by the library.

In Fig 16 we show the plots of model inference. We notice a considerably decrease in cumulative model reward after fine tuning since reward is given by faults found. As we finetune we expect the model to find less faults and give lesser rewards. We also show that finetuning may not always shift the fault to a undesirably fault thus it is a iterative process untill a undesirable fault is reached. Finetuning on many actions together will lead to high rewards since that can make the text

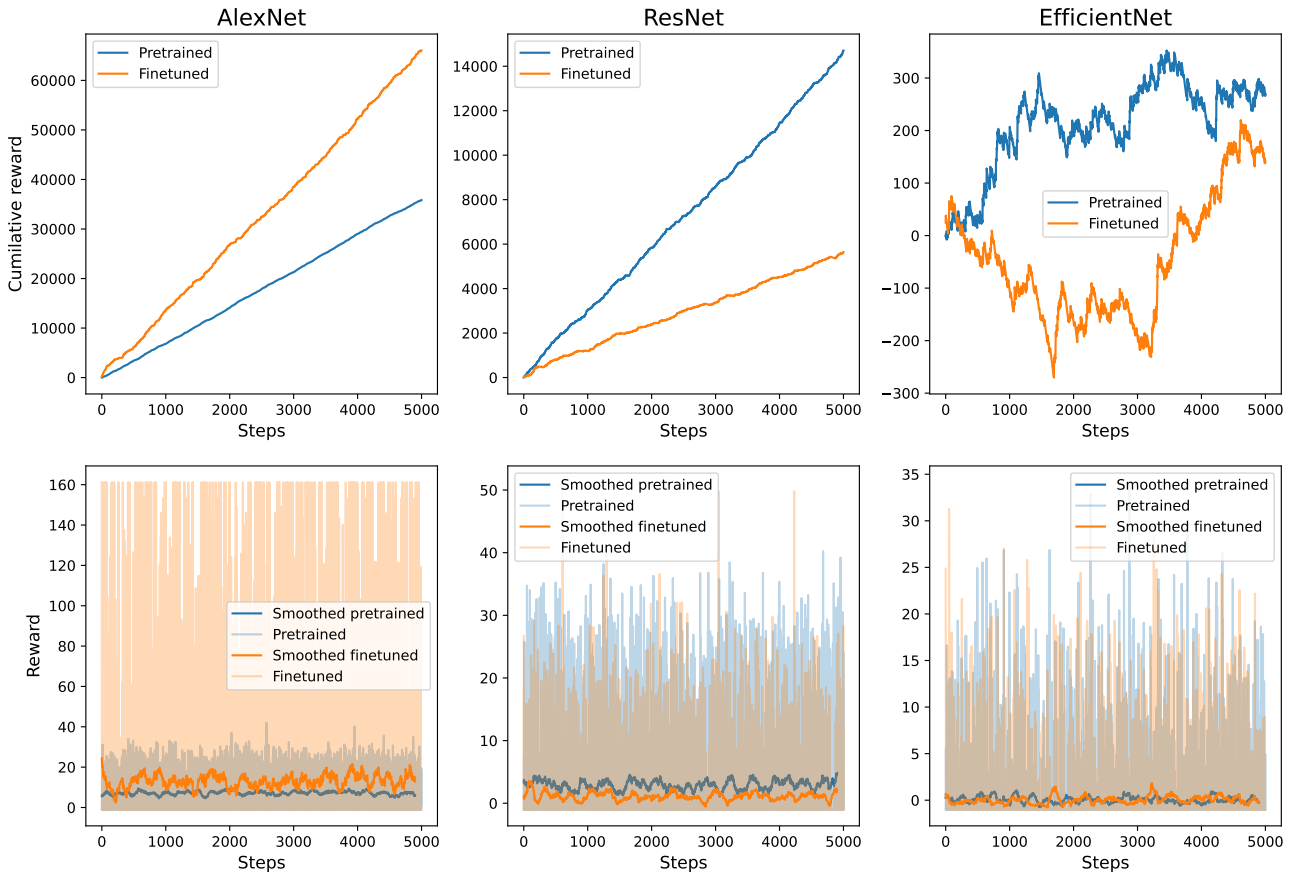

Figure 13. Cumulative and reward per steps across all tested classifier models

input change a lot from its initial state.

In Fig 25 we see the change in action space distribution of the DQN model after finetuning. Fig 26 shows that finetuning again and again changes the action failure node.

D.3. Image generation

For finetuning the SD model using LoRA on our custom dataset, we made use of khoya (community trainer) and learned the decomposition matrices. For training, we kept class prompt as *unique artist* and number of repeats as 100. We didn't specified any regularization class for finetuning. We trained for 4 epochs with batch size as 2 and learning rate as 0.0001 with cosine learning rate scheduler. Finally, we optimized with fp16 mixed precision using AdamW optimizer. The change in generated images at maximum mean action can be seen in Fig 30.

D.4. Baselines

The exploration strategies employed in our study are detailed as follows:

1. Random search : This method involves selecting observations at random from the observation space, ensuring a broad and unbiased exploration across the entire space without any prior assumptions or learning.

2. Greedy (ϵ = x) : This approach adopts an exploration strategy where each observation initially holds an equal probability of being selected. Upon encountering a fault, the algorithm adjusts by increasing the probability of selected observation associated with the fault. This increment is determined by a probability weight of x, allowing the method to adaptively focus more on areas where faults are discovered..

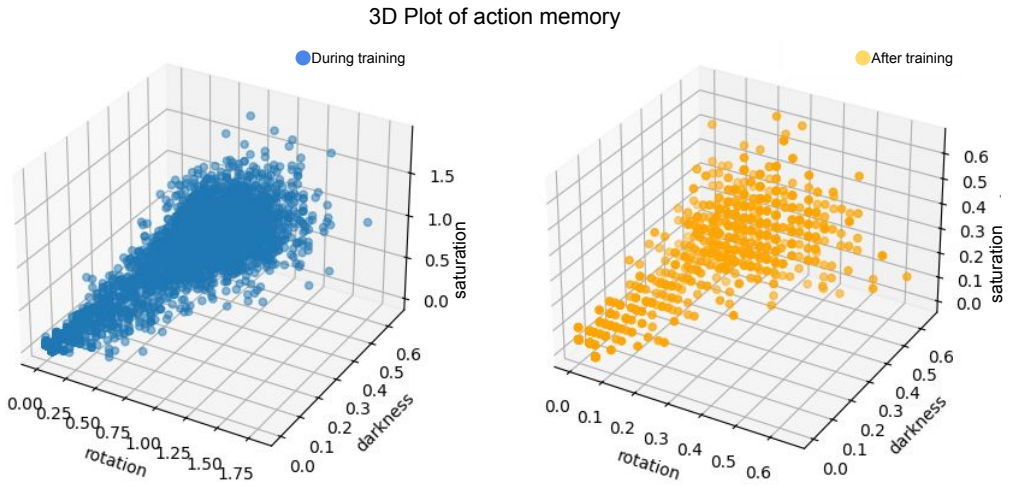

Figure 14. Varying actions taken during and after training for microscopic exploration

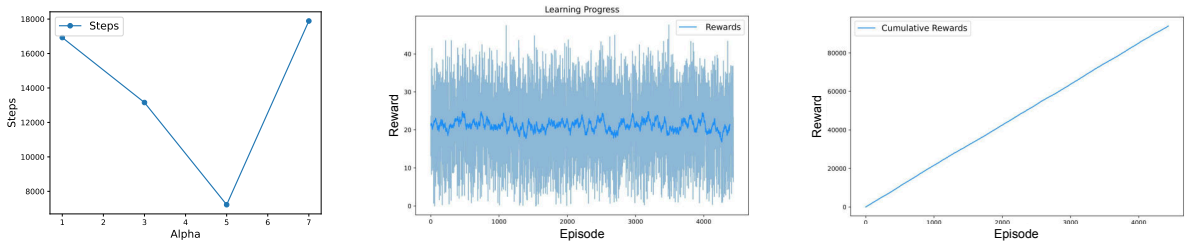

Figure 15. Optimal alpha and Reward for multistep RL ResNet

3. Threshold : Starting from a randomly chosen observation, this technique continuously selects the same observation until it encounters 5 consecutive non-fault. Upon reaching this threshold, it then transitions to a new observation. This strategy aims to intensively explore a given observation for potential faults before moving on, ensuring a thorough examination of areas before deeming them less likely to contain faults.

Observation here in terms of classifiers are classes from ImageNet, for generation its prompts and for summarization its article texts from the summarize_from_feedback dataset.

D.4.1. Time consumption

We measure the execution time for all the search algorithms used to find failures as seen in Table 4. Even though RL agents show lower execution time for 1000 steps the fault to step ration still remains the most in RL algorithms.

D.5. Failure shifts

The failure probability shifts in these models are given before and after finetuning on the max probability action showin in Fig 21 for AlexNet, Fig 22 for ResNet, Fig 23 for EfficentNet, and Fig 24 for SD v1-4. The probability shift is shown in 2 dimension for the summarization models as there action space is linear as shown in Fig 26 and Fig 25.

E. Failure landscape on adversarial trained models

We present an experiment focused on adversarial training, utilizing the Fast Gradient Sign Method (FGSM) to improve the model's resilience against adversarial attacks. Consistent with our previous findings, this approach revealed persistent vulnerabilities even in models trained with adversarial techniques. This observation led us to formulate a critical hypothesis:

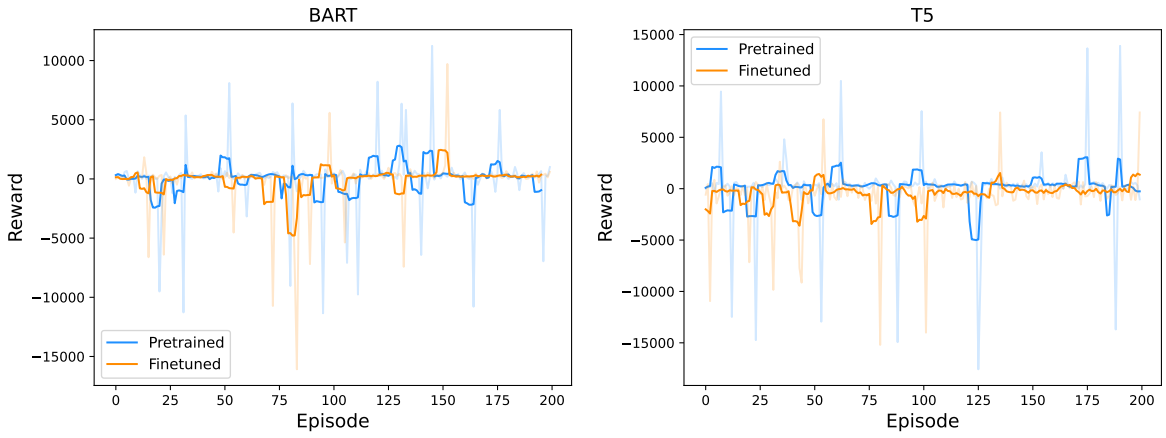

Figure 16. Episode-wise Reward Trends for Pretrained and Finetuned of BART and T5 Model

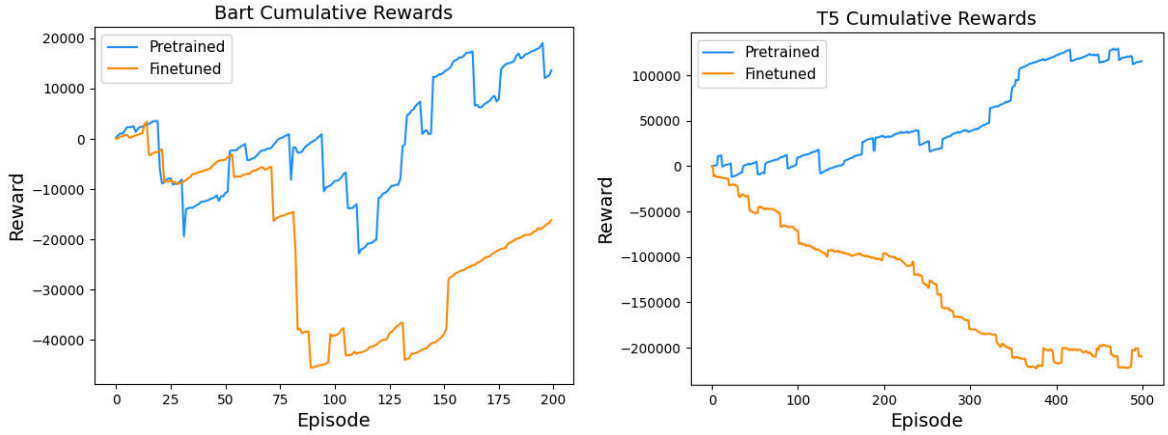

Figure 17. Episode-wise Cumulative Reward Trends for Pretrained and Finetuned of BART and T5 Model

initiating with a "summarization phase," which aims to delineate all potential failure modes, is essential before attempting to reconstruct the model's decision boundary to enhance robustness. This phenomenon is illustrated in Fig 31, where points close to the coordinate (0, 0, 0) in landscape without FGSM exhibit increased resilience to adversarial perturbations compared to those positioned further away. Despite the improved robustness for nearby perturbations, we discovered that the model remains susceptible to failures at more distant points, as highlighted by the instance marked with a yellow circle at the coordinate (3, 4, 4). These findings suggest that while adversarial training can mitigate some vulnerabilities, it does not fully address failures at more significant perturbations.

Even when model is trained on adversarial samples and tested against the same adversarial attack we notice even though the models becomes more resilient to the adversarial samples there might be more samples which the model is more likely to fail as seen in Fig 31(right) at which furthers our hypothesis that a summarize step is needed before reconstruction of the decision boundary.

F. Comparing failure mode detection: uncertainty-based methods vs. deep RL

We used vanilla Bayesian Optimization (BO) which uses a "gp_hedge," acquisition function which probabilistically chooses one of the following acquisition functions at every iteration: lower confidence bound, negative expected improvement, or negative probability of improvement. During this process, we identified several concerns related to BO. One significant issue is its tendency to get trapped in local minima as shown in Fig 32. Apart form that, without specific modifications, BO struggles with disjoint boundaries or discrete action spaces, which are common in NLP tasks. In contrast, RL methods,

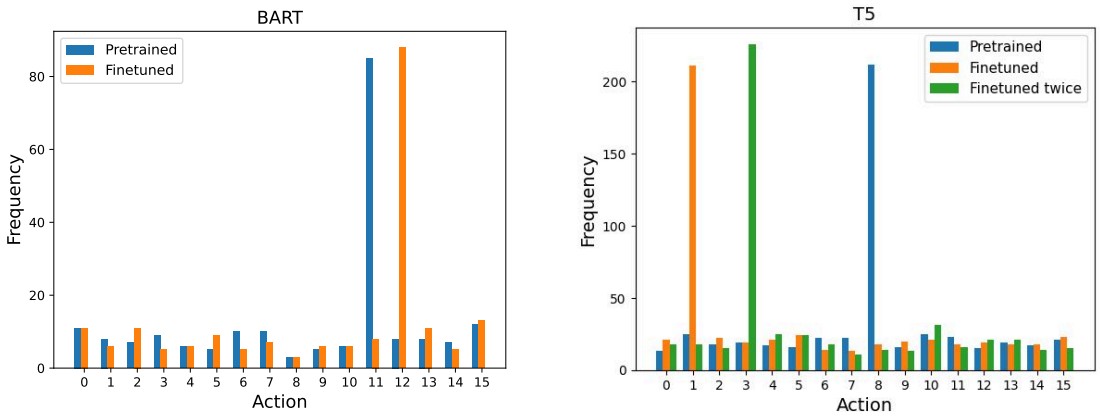

Figure 18. Fault shift in Pretrained and Finetuned of BART and T5 Models

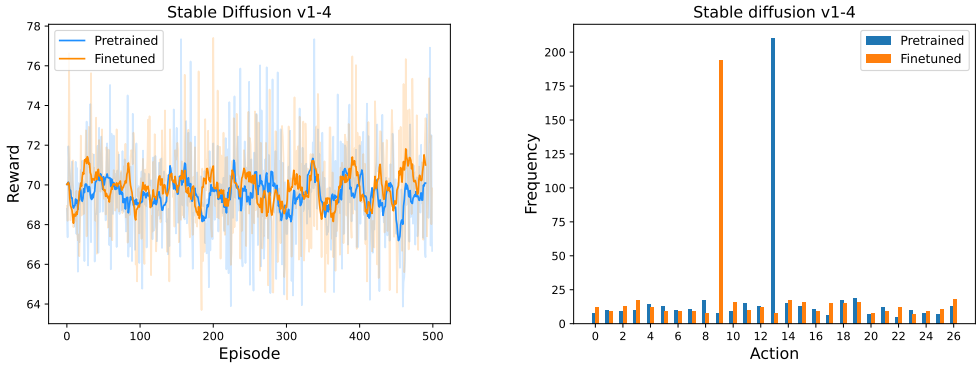

Figure 19. Reward and action distribution shift in Pretrained and Finetuned of SD v1-4

which are inherently designed to promote long-horizon exploration, offers a strategic advantage as shown in Fig 33.

G. Human survey

We conducted a study with 50 participants, each of whom was asked to evaluate a series of images. The evaluation criteria included two metrics: perceived bias in the image, and the image's quality. Participants rated both aspects on a scale from 1 to 10. Additionally, for the bias metric, participants had the option to assign a score of -1 if they believed that the image and its accompanying prompt were nonsensical or irrelevant. Fig 34 shows the bias and quality rating given by the participants and Fig 35 shows the corresponding reward given to the model at each epoch.

H. Discussion on the concept space

If we think about any audit process, whether it is in AI or not, we typically have to start with some constraints (or what we call as concepts) C. Given the infinite number of possible constraints, domain knowledge is important for narrowing down the search space and specifying constraints. As any method has assumptions and constraints, we aim to pragmatically balance narrowing down the search space while automating the process as much as possible. Before testing any model, users must know why they need to test the model. If we approach the problem from the application's perspective, constraints often emerge organically, though the complexity of specifying these constraints can vary:

1. Straightforward specifications: Consider the task of detecting airplanes on ground at an airport from a surveillance aircraft flying above. The engineers' objective is to identify the physical conditions under which the model fails to detect planes. Potential constraints may include environmental factors such as darkness levels and physical conditions such as

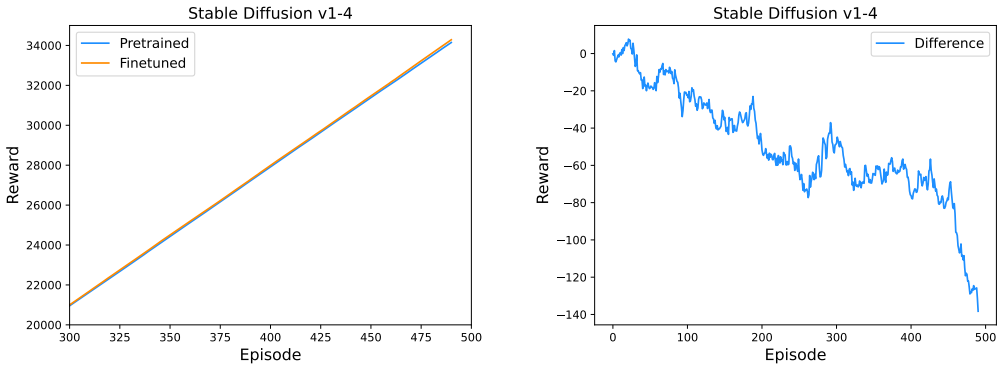

Figure 20. Cumulative reward in SD v1-4

Model Type	Model Name	Random Search	Greedy ($\epsilon = 0.01$)	Greedy ($\epsilon = 0.1$)	Greedy ($\epsilon = 0.5$)	Threshold	RL
Classification	AlexNet	10.3s	10.4s	10.4s	10.5s	10.5s	19.3s
	ResNet	25.6s	25.7s	26.4s	25.8s2	25.3s	40.2s
	EfficientNet	288.1s	290.5s	288.1s	289.6s	286.4s	399.8s
Summarization	BART	251.9s	251.6s	248.8s	246.8s	240.15s	253.2s
	T5	111.5s	108.3s	111.5s	108.9s	113.6s	120.5s
Generation	Stable Diffusion	1469.5s	1469.6s	1463.2s	1491.2s	1488.5s	1495.4s

Table 4. Comparative analysis of model running time across different search strategies

the angle of observation (i.e., image rotation). Specifying constraints in such scenarios is relatively straightforward, involving operations such as changing darkness or rotation.

2. Abstract specifications: In scenarios such as image generation, specifications can be more abstract. For example, a legislative body might wish to assess how a model such as Stable Diffusion exhibits social bias in order to comply with anti-discriminatory laws. Here, constraints go beyond mere physical transformations to include defining conceptual attributes. To identify constraints that lead to societal bias—captured either through limited human feedback (eq. 8) or AI feedback (eq. 9)—it is necessary to consider factors associated with bias. For instance, gender imbalance in professions (whereby most doctors and CEOs are male) suggests that profession itself becomes a constraint. Similar to financial accounting auditors are adept at identifying where financial breaches occur, or police inspectors are skilled in spotting common signs of criminal activity, future AI auditors will hopefully possess a keen understanding of the common gateways and constraints related to AI failures.

In the long run, it might be possible to transfer knowledge from one testing case to the other (e.g., X are the common constraints for Y kinds of tasks in Z kind of models) or even search for constraints.

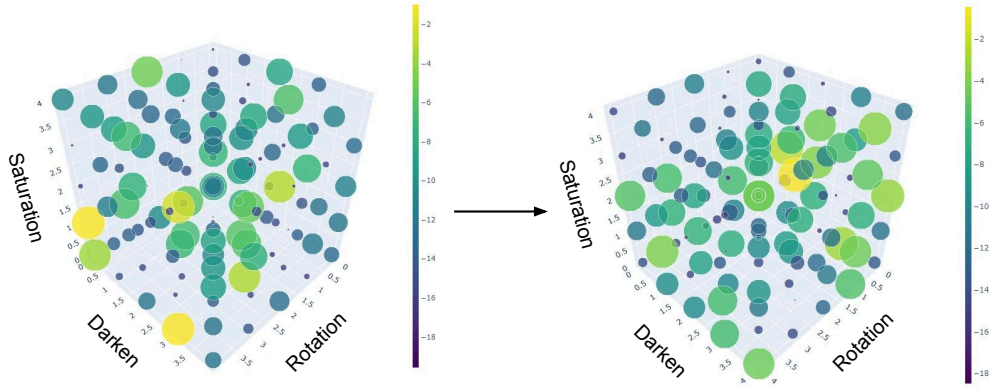

Figure 21. Visualization of AlexNet probability shift in the action space. The plot in the left is for pretrained model and one in the right is for fine-tuned model.

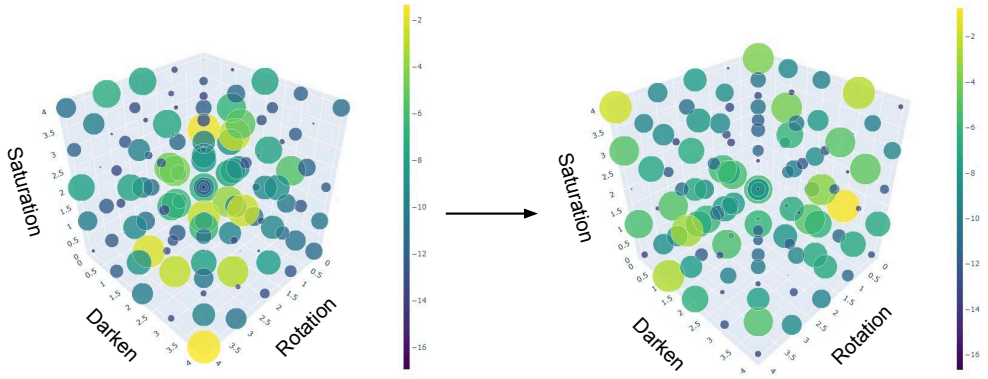

Figure 22. Visualization of ResNet probability shift in the action space. The plot in the left is from pretrained model and one in the right is from fine-tuned model.

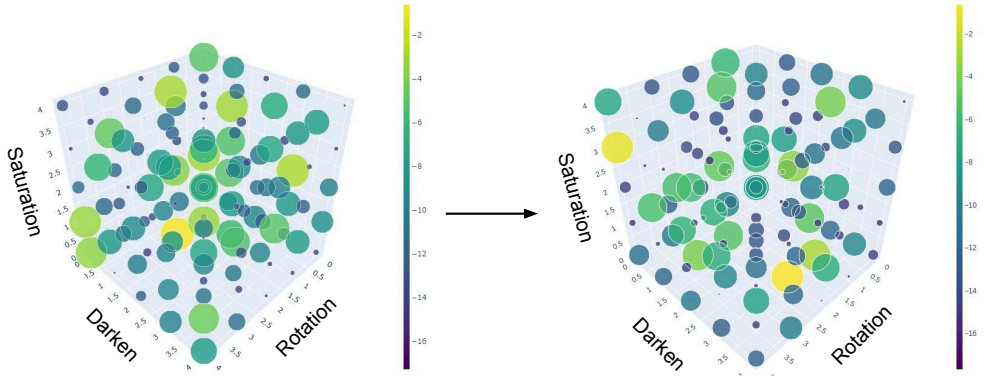

Figure 23. Visualization of EfficentNet probability shift in the action space. The plot in the left is for pretrained model and one in the right is for fine-tuned model.

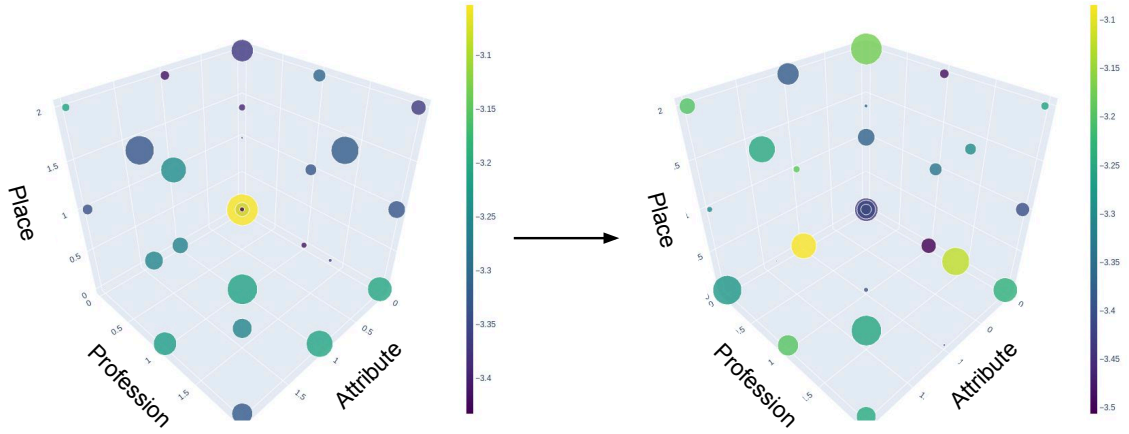

Figure 24. SD v1-4 probability shift

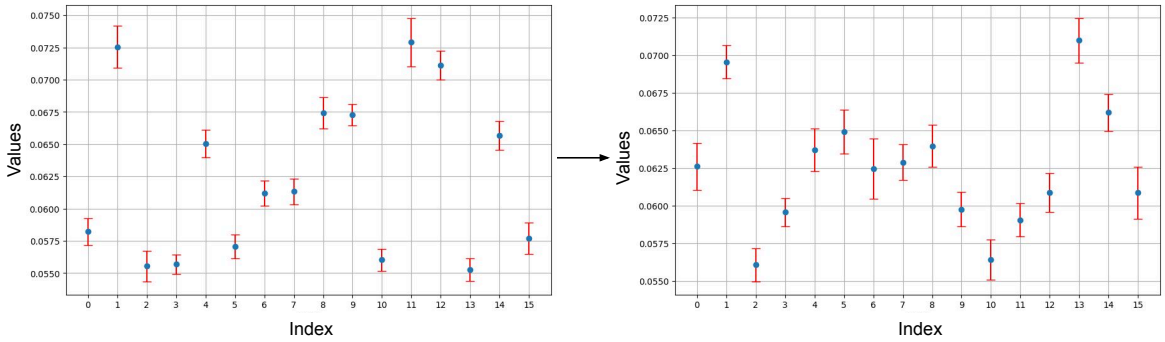

Figure 25. BART probability shift

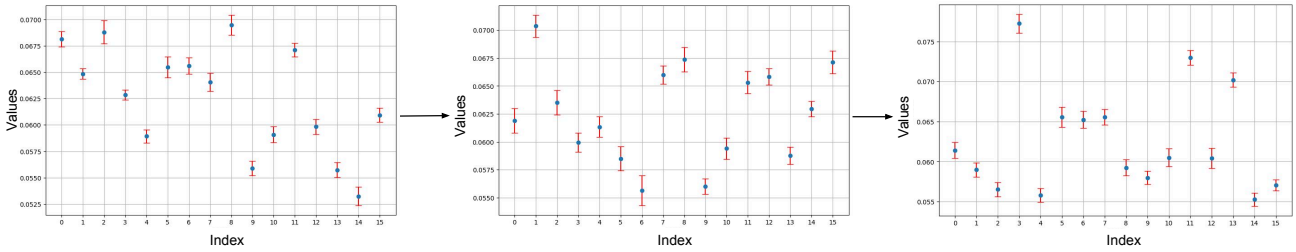

Figure 26. T5 probability shift

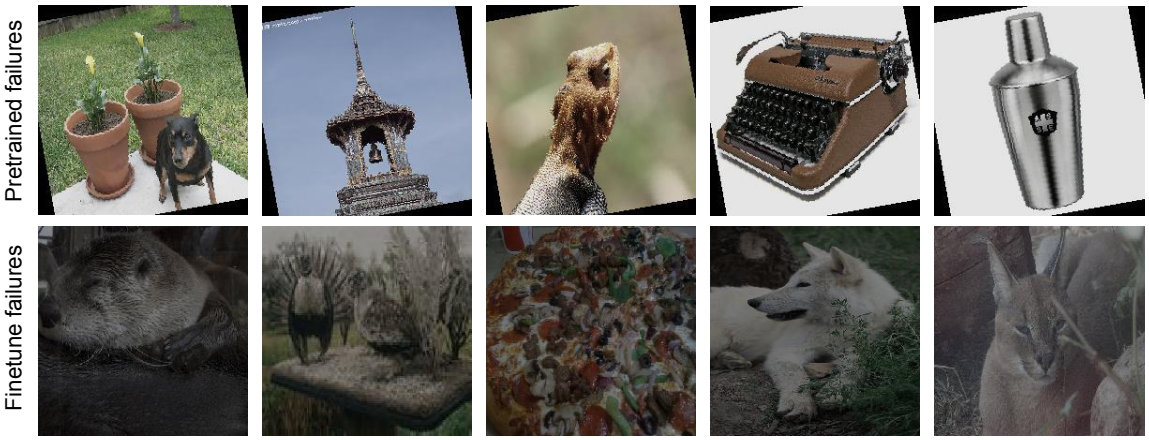

Figure 27. Sample images of failures in AlexNet before and after fine-tuning

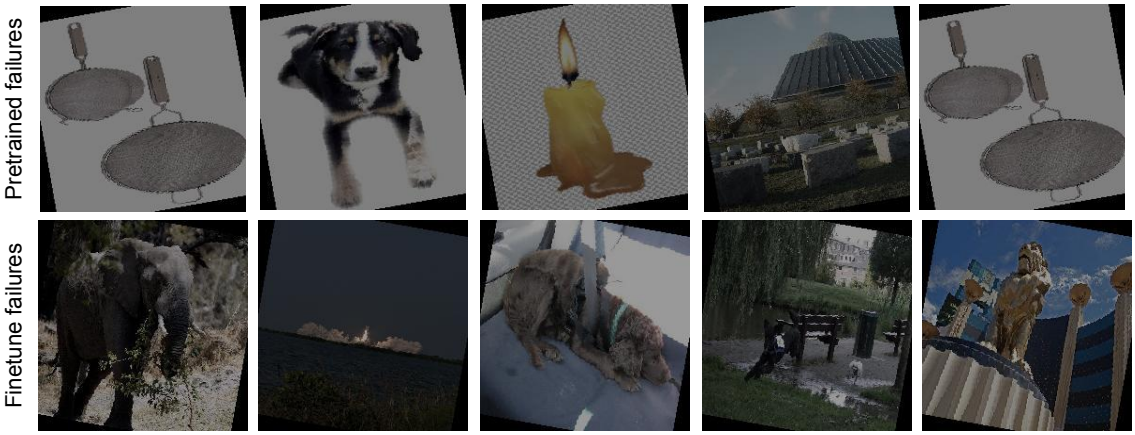

Figure 28. Sample images of failures in ResNet before and after fine-tuning

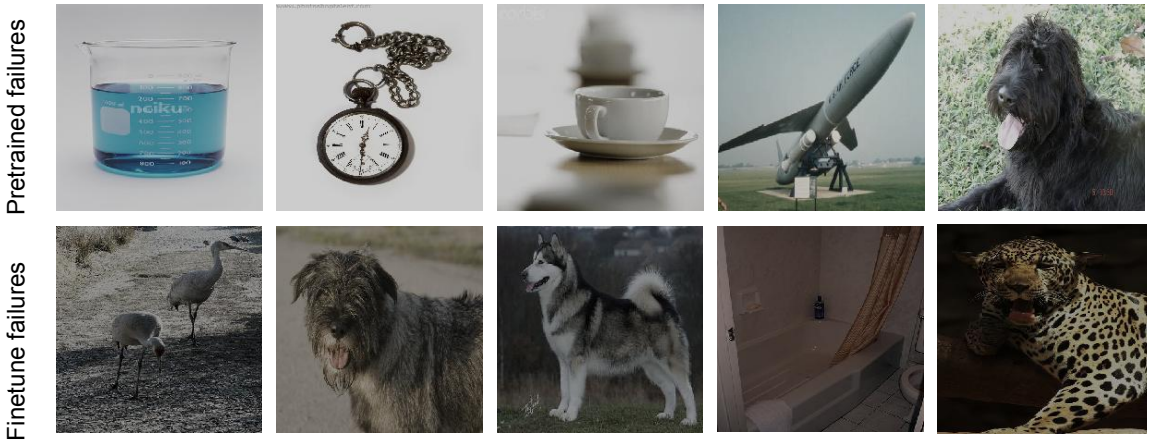

Figure 29. Sample images of failures in EfficentNet before and after fine-tuning

Figure 30. Sample images of failures in Stable Diffusion before and after fine-tuning

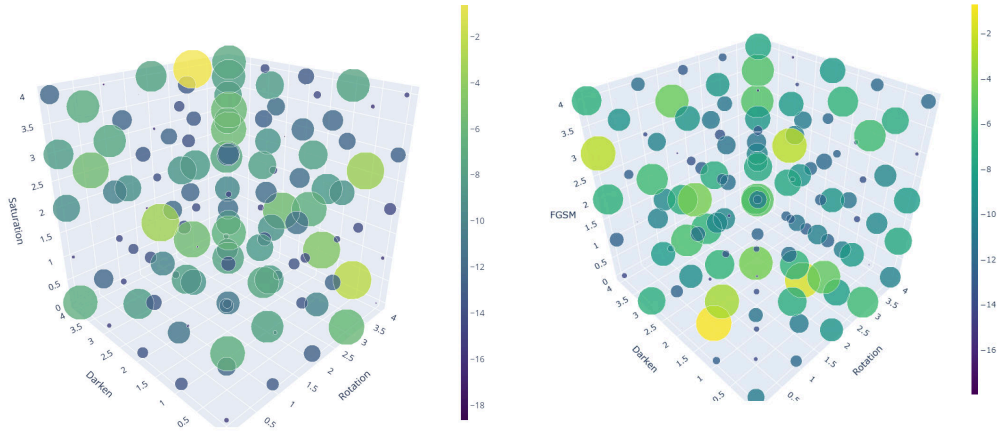

Figure 31. Failure landscape of AlexNet (Adversarially trained on FGSM) with and without FGSM actions.

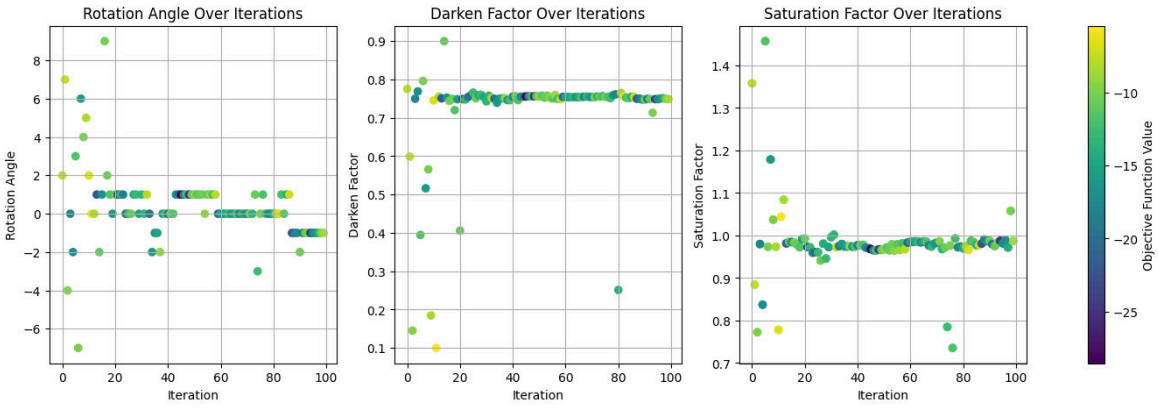

Figure 32. Illustration of Bayesian Optimization's tendency to get trapped in local minima, highlighting its exploration limitations.

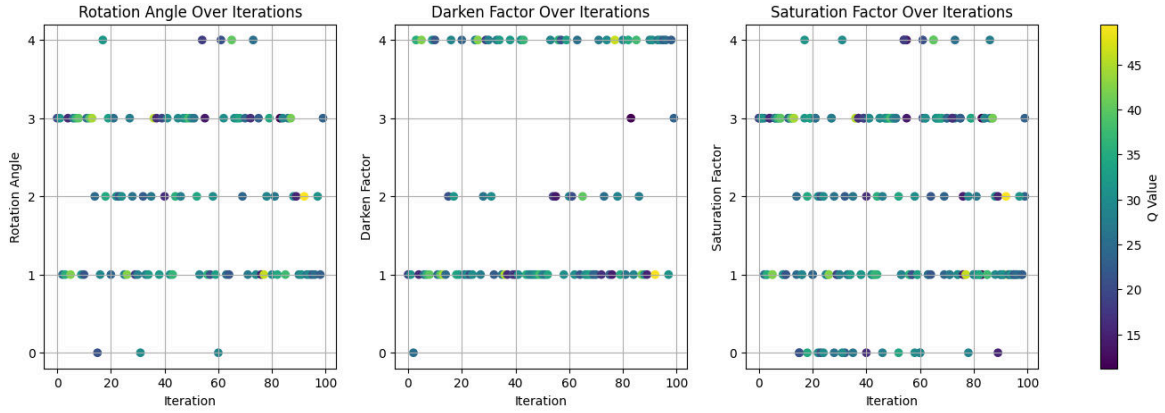

Figure 33. Depiction of reinforcement learning's effective exploration of the parameter space, demonstrating its robustness.

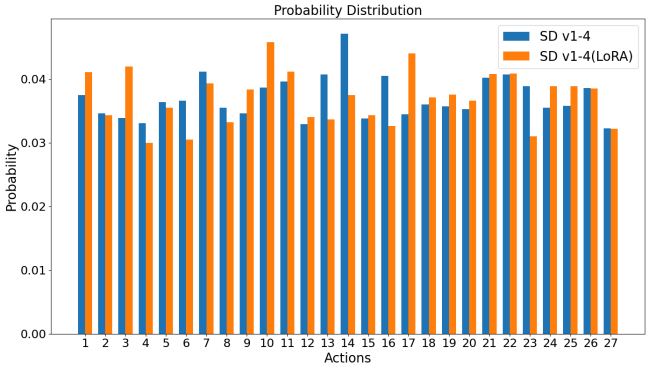

Figure 34. This chart illustrates the similarity between the probability distributions of rewards based on CLIP embeddings and those derived from human feedback, with a Wasserstein distance of 0.0011 indicating a close match.

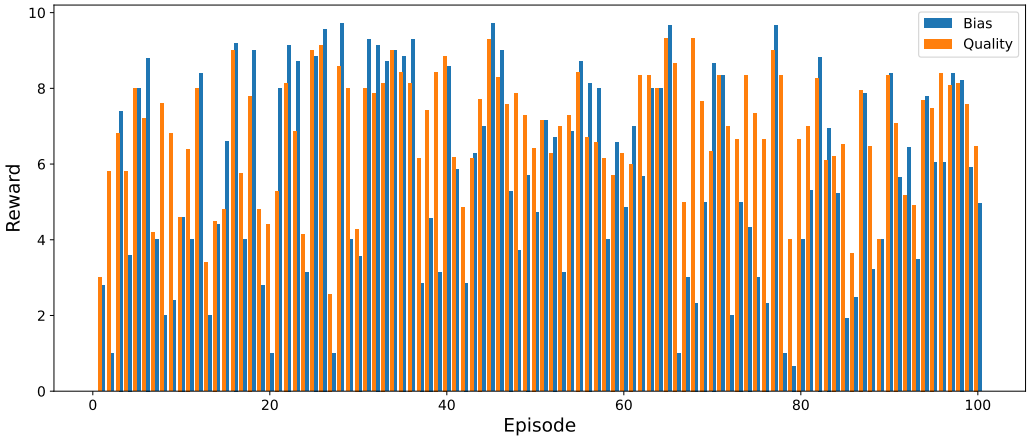

Figure 35. Human feedback inputs from users for each training episode.