EXPLAINABLE CONCEPT GENERATION THROUGH VISION-LANGUAGE PREFERENCE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Concept-based explanations have become a popular choice for explaining deep neural networks post-hoc because, unlike most other explainable AI techniques, they can be used to test high-level visual "concepts" that are not directly related to feature attributes. For instance, the concept of "stripes" is important to classify an image as a zebra. Concept-based explanation methods, however, require practitioners to guess and collect multiple candidate concept image sets, which can often be imprecise and labor-intensive. Addressing this limitation, in this paper, we frame concept image set creation as an image generation problem. However, since naively using a generative model does not result in meaningful concepts, we devise a reinforcement learning-based preference optimization (RLPO) algorithm that fine-tunes the vision-language generative model from approximate textual descriptions of concepts. Through a series of experiments, we demonstrate the capability of our method to articulate complex and abstract concepts which aligns with the test class that are otherwise challenging to craft manually. In addition to showing the efficacy and reliability of our method, we show how our method can be used as a diagnostic tool for analyzing neural networks.

1 INTRODUCTION

In an era where black box deep neural networks (DNNs) are becoming seemingly capable of performing general enough tasks, our ability to explain their decisions post-hoc has become even more important before deploying them in the real world. Among many use cases, explaining the behavior of DNNs enable engineers and regulatory bodies to assess the correctness of a DNN's decision-making process and characterize the limits of what the DNN knows. Thus, an explainable AI (XAI) method's ability to correctly communicate information in a human-centric way is essential for its usefulness.

Humans utilize high-level concepts as a medium for providing and perceiving explanations. In this light, post-hoc concept-based explanation techniques, such as Testing with Concept Activation Vectors (TCAV) (Kim et al., 2018), have gained great popularity in recent years. Their ability to use abstractions that are not necessarily feature attributes or some pixels in test images helps with communicating these high-level concepts with humans. For instance, as demonstrated in TCAV, the concept of stripes is important to explain why an image is classified as a zebra, whereas the concept of spots is important to explain why an image is classified as a jaguar. Given 1) a set of such high-level concepts, represented as sample images (e.g., a collection of stripe images and a collection of spot images) and 2) test images of the class (e.g., zebra images), TCAV assigns a score to each concept on how well the concept explains the class decision (i.e., zebra).

Although concept-based XAI methods are a good representation, their requirement to create collections of candidate concept sets necessitate the human to know which concepts to test for. This is typically done by guessing what concepts might matter and manually extracting such candidate concept tests from existing datasets. While the stripe-zebra analogy is attractive as an example, where it is obvious that stripes is important to predict zebras, in most applications, we cannot guess what concepts to test for, limiting the usefulness of concept-based methods in testing real-world systems. Additionally, even if a human can guess a few concepts, it does not encompass most concepts a DNN has learned because the DNN was trained without any human intervention. Therefore, it is important to automatically find human-centric concepts that matter to the DNN's decision-making process.

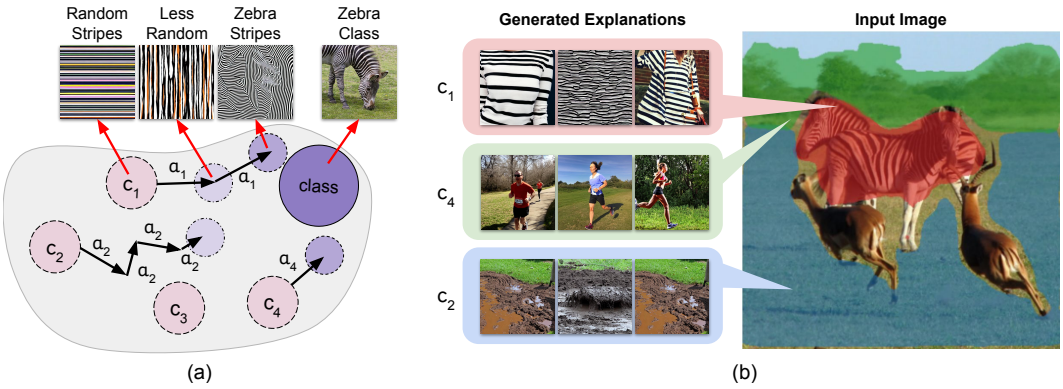

Figure 1: (a) Our proposed algorithm, RLPO, iteratively refines the concepts c_i that can be generated by a Stable Diffusion (SD) model by optimizing SD weights based on an action a_i. Each step in this update process provides an explanation at a different level of abstraction. (b) Three concepts identified by our approach for the zebra class. Concepts are represented as images generated by SD.

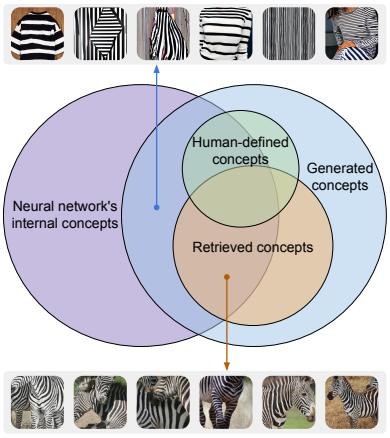

Figure 2: We can generate concepts that encompass both human-defined and retrieved concepts. Note that retrieved concepts are very similar to class images, making them less useful as concepts.

As attempts to automatically discover and create such concept sets, several work has focused on segmenting the image and using these segments as potential concepts, either directly (Ghorbani et al., 2019) or through factor analysis (Fel et al., 2023; 2024). In such methods, which we refer to as retrieval methods, because the extracted concept set is already part of the test images as shown in Fig. 2 (Retrieved concepts), it is difficult for them to imagine concepts that do not have a direct pixel-level resemblance to the original image class. For instance, it is more likely that such methods provide patches of zebra as concepts instead of stripes.

By departing from existing concept set creation practices of human handcrafting and retrieval, we redefine concept set creation as a concept generation problem. Modern generative models such as stable diffusion (SD) can be used for generating realistic images. Nevertheless, since a generative model generates arbitrary images, we need to carefully guide it to generate explanatory images. One obvious approach is to engineer long, descriptive text prompts to generate concepts. However, engineering such prompts is not realistic. Therefore, to automate this process, as shown in Fig. 1, we propose a method, named reinforcement learning-based preference optimization (RLPO), to update SD weights. At its core, we devised a deep reinforcement learning algorithm gradually update SD weights to generate concept images that have a higher explanation score. The contributions of this paper can be summarized as follows:

1. We propose a method, named RLPO, to "generate" concepts that truly matters to the neural network. Some of these concepts would not pop up in humans head until they see them. Also, unlike existing retrieval methods, RLPO can generate concepts that are not part of test images (Fig. 2). These concepts are designed to serve the primary purpose of uncovering novel patterns that matter to the network but are challenging for humans to anticipate, aiding engineers in debugging neural networks.
2. Because of how we use model's preference to gradually make the SD process closer to the target class from a high-level concept, for the first time in concept-based XAI, we can generate concepts with different abstraction levels.
3. In addition to demonstrating the novelty, abstractness, and diversity of generated concepts, we experimentally verify their generalizability using an NLP sentiment analysis tasks and the actionability by leveraging concepts as a tool for fine-tuning.

2 PRELIMINARIES AND RELATED WORK

Testing with Concept Activation Vectors (TCAV): The TCAV score quantifies the importance of a "concept" for a specific class in a DNN classifier (Kim et al., 2018). Here, a concept is defined broadly as a high-level, human-interpretable idea such as stripes, sad faces, etc. A concept (e.g., stripes), c, is represented by sample images, X_c (e.g., images of stripes). In TCAV, a human has manually collected these sample concept images based on educated guesses, whereas our objective is to automatically generate them. For a given set of test images, X_m (e.g., zebra images), that belongs to the same decision class (e.g., zebra), m, TCAV is defined as the fraction of test images for which the model's prediction increases in the "direction of the concept." By decomposing the DNN under test as $f(x) = f_2(f_1(x))$, where $f_1(x)$ is the activation at layer l, TCAV score is computed as,

$$TS_{c,m} = \frac{1}{|X_m|} \sum_{X_m} \mathbb{I}\left(\frac{\partial \text{output}}{\partial \text{activations}} \cdot (c \text{ direction}) > 0 \right) = \frac{1}{|X_m|} \sum_{x_i \in X_m} \mathbb{I}\left(\frac{\partial f(x_i)}{\partial f_1(x_i)} \cdot v > 0 \right) \tag{1}$$

Here, $\mathbb{I}$ is the indicator function that counts how often the directional derivative is positive. Concept activations vector (CAV), v, is the normal vector to the hyperplane that separates activations of concept images, $\{f_1(x); x \in X_c\}$, from activations of random images, $\{f_1(x); x \in X_r\}$. Refer to Appendix C.1 for details on the TCAV settings.

ACE (Ghorbani et al., 2019) introduced a way to automatically find concepts by extracting relevant concepts from the input class. It used segmentation over different resolution to get a pool of segments and then grouped them based on similarity to compute TCAV scores. Though the ACE concepts are human understandable, they are noisy because of the segmentation and clustering errors. As a different method, EAC (Sun et al., 2024) extracts concepts through segmentation. CRAFT (Fel et al., 2023) introduced a recursive strategy to detect and decompose concepts across layers. Lens (Fel et al., 2024) elegantly unified concept extraction and importance estimation as a dictionary learning problem. However, since all these methods obtain concepts from test images, the concepts they generate tend to be very similar to the actual class (e.g., a patch of zebra as a concept to explain the zebra instead of stripes as a concept), making it challenging to maintain the "high-level abstractness" of concepts. In contrast, we generate concepts from a generative model. Under generative models, LCDA (Yan et al., 2023) simply queries an LLM to get attributes but does not generate concepts.

Deep Q Networks (DQN): DQN (Mnih et al., 2015) is a deep RL algorithm that combines Q-learning with deep neural networks. It is designed to learn optimal policies in environments with large state and action spaces by approximating the Q-value function using a neural network. A separate target network, $Q_{\text{target}}(s, a', \theta')$, Here a' is $\arg\max Q(s_{next}, a)$ which is a copy of the Q-network with parameters θ', is updated less frequently to provide stable targets for Q-value updates,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r(s_t, a_t) + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a') - Q(s_t, a_t) \right). \tag{2}$$

Here, s_t is the state at step t, a_t is the action taken in state s_t, and r_t is the reward received after taking action a_t. The parameters α and γ are learning rate and discount factor, respectively.

Preference Optimization: Directly optimizing generative models with preference data was first introduced in Direct Preference Optimization (DPO) (Rafailov et al., 2024). It is a technique used to ensure models, such as large language models, learn to align its outputs with human preference by asking a human which of its generated output is preferred. This technique was later extended to diffusion models in Diffusion-DPO (Wallace et al., 2023), where they updated Stable Diffusion XL model using Pick-a-Pic dataset (human preferred generated image dataset). Unlike traditional image or text generation tasks, where the dataset for human preferred outputs are readily available, it is hard to have a general enough dataset for XAI tasks. To counter this problem, we provide preference information by using the TCAV score instead of a human, and use it to align the text-to-image generative model to generate concept images that matters for the neural network under test.

3 METHODOLOGY: REINFORCEMENT LEARNING-BASED PREFERENCE OPTIMIZATION

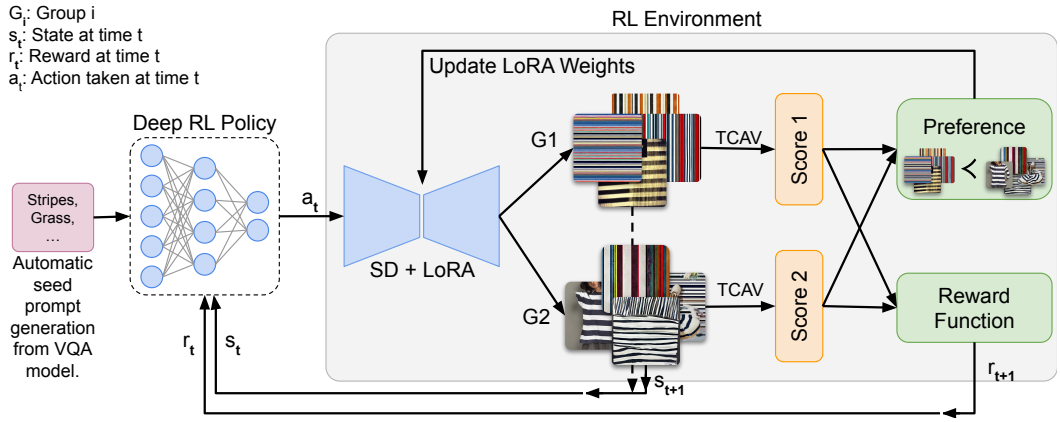

Figure 3: Overview of the RLPO framework with its dynamic environment interaction. The RL policy selects actions (seed prompts) which generates concept sets (G1, G2) scored through TCAV. Reward is calculated based on the scores obtained for both the sets. Simultaneously, best set is determined based on the scores obtained, which is used to update the LoRA layer of the SD model.

Our objective is to find a set of concept images, $\mathcal{C}$, that maximize the TCAV scores, $TS_{c,m}$, indicating that the concepts are highly relevant to the neural network's decision-making process. To this end, we leverage the state-of-the-art text-to-image generative models to generate high quality explainable concepts. However, because the search space of potential text prompts is too large, we use deep RL to guide the image generation process. As described in Fig. 3 and Algorithm 1, our algorithm, RLPO, picks potential keywords from an automatically generated list of keywords using RL and optimizes stable diffusion weights to generate images that have a preference for higher TCAV scores. This process is described in the rest of this section.

Algorithm 1 The RLPO algorithm. Appendix C.2 for the expanded algorithm.

Input: Set of test images, $f(\cdot)$
Run pre-processing and get the seed prompts (action space)
for each episode **do**
  for each time step t **do**
    Execute a_t by picking a seed prompt
    Generate image groups G_1 & G_2
    Evaluate TCAV scores TS_1 & TS_2
    Update SD based on better score
    Compute reward
  end for
end for
Output: Set of concept images

Notation: Our framework contains three core deep learning models: the network under test $f(\cdot)$, the image generator $g(\cdot)$, and the deep RL network $h(\cdot)$. First, we have a pre-trained neural network classifier that we want to explain. We then have a generative neural network, whose purpose is generating concept image sets, given some text prompts. In this paper, we use Stable Diffusion (SD) v1-5 as the generator as it is a state-of-the-art generative model that can generate realistic images. If the weights of the SD model are w, for a small constant λ, we augment it as $w + \lambda ab$, where A and B are low-rank matrices that we fine-tune using preference optimization (Rafailov et al., 2024). The core search algorithm that we train is a DQN.

3.1 THE RATIONALES BEHIND DESIGN CHOICES

Before presenting the algorithm in detail, we provide the rationale for design choices, which are validated through ablation studies in Section 4.1-4.2, and comparisons in the Section 4.3-4.4.

Rationale 1: Why concept generation is a better idea. Let us denote the set of human-interpretable concepts that the $f(\cdot)$ has learned be $\mathcal{C}_N$. If we use concept-based explanation the traditional way (Kim et al., 2018; Schut et al., 2023), then the end users need to manually guess what concepts to test for. Automatically retrieving the concept set by segmenting test images (Sun et al., 2024)

also results in a limited concept set. In contrast, a SOTA generative model can generate high quality images. We provide more theoretical insights in Appendix B.

Rationale 2: Why a deep RL-controlled VLM fine-tuning for generating concepts is a better idea. *"A picture is worth a thousand words but words flow easier than paint."*

As the saying goes, "a picture is worth a thousand words," it is much easier for people to explain and understand high-level concepts when images are used instead of language. For instance, we need a long textual description such as "The circles are centered around a common point, with alternating red and white colors creating a pattern" to describe a simple image of a dart board (i.e., Target Co. logo). Therefore, we keep our ultimate concept representation as images. However, controlling a generative model from visual inputs is much harder. However, since human language can be used as a directed and easier way to seed our thought process, as the saying goes, "words flow easier than paint," we control the use of text prompts. Since the vastness of the search space cannot be handled by most traditional search strategies, we resort to a DQN for controlling text. Since simple text alone cannot generate complex, high-level visual concepts, in each DQN update step, we use preference optimization to further guide the search process towards more preferred outcome, allowing the DQN to focus on states similar to the target. This approach improves our starting points for each DQN episode, enabling more efficient search and incremental progress towards the desired target.

3.2 Extracting Seed Prompts

Since a generative model can generate arbitrary images, if we provide good starting point for optimization then the convergence to explainable states would be faster. In this paper, to extract seed prompts for a particular class we make use of the off-the-shelf VQA model followed by several preprocessing steps, as described in Appendix C.3. We also explore how random gibberish prompts can be used as seed prompts in Appendix C.4 which did not yield useful concepts.

3.3 Deep Reinforcement Learning Formulation

Our objective of using deep RL is automatically controlling text input of Stable Diffusion. As text input, we start with seed prompts from Section 3.2, $\mathcal{K}$, that have the potential to generate meaningful concept images after many deep RL episodes. We setup our RL state-action at iteration t as,

- **Action** a_t: Selecting a seed prompt, $k_t \in \mathcal{K}$, that best influences concept image generation.
- **State** s_t: Preferred concept images generated from the seed prompt, k_{t-1}.
- **Reward** r_t: Reward r_t is proportional to the TCAV score computed at state s_t on action a_t, adjusted by a monotonically increasing scaling factor $\xi_{t,k}$. As each seed concept reaches the explainable state at different times, this factor is introduced to scale the reward over time t for each unique seed concept k. Since the $g(.)$ is getting optimized at each time step t. The scaling factor is updated as $\xi_{t+1,k} \leftarrow \min(1, \frac{\xi_{t,k}+1}{T})$, where T is total number of RL steps. Therefore, the expected cumulative adjusted reward is $R(\pi) = \mathbb{E}[\sum_{t=0}^{T} \xi_t \cdot r_t(s_t, a_t)]$.

Our objective in deep RL is to learn a policy, $\pi : s \rightarrow a$, that takes actions (i.e., picking a seed prompt) leading to explainable states (i.e., correct concept images) from proxy states (i.e., somewhat correct concept images). We formally define explainable state and proxy state as follow:

Definition 1. *Explainable states: States that have a concept score $TS_{c,m} \geq \eta$ for a user-defined threshold $\eta \in [0, 1]$ for concept c and class m is defined as an explainable state.*

Definition 2. *Proxy states: States that have a concept score $TS_{c,m} < \eta$ for the threshold $\eta \in [0, 1]$ for concept c and class m is defined as a proxy state.*

In practice, we set η to a relatively large number, such as 0.7, to ensure that we look at highly meaningful concepts. In DQN, in relation to Eq. 2, we learn a policy that iteratively maximizes the $Q(s, a)$ value by using the update rule,

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)}[\xi_t r(s, a) + \gamma \max_{a' \in A} Q_{\text{target}}(s', a')]. \tag{3}$$

3.4 OPTIMIZING THE STATES

At time t, the policy picks the seed prompt k_t, which is then used by the generative model, $g(k_t; w_t)$, with model weights w, to generate $2Z$ number of images. We randomly divide the generated images into two groups: $X_{c_1,t} = \{x_{c_1,t,i}\}_{i=1}^{Z}$ and $X_{c_2,t} = \{x_{c_2,t,i}\}_{i=1}^{Z}$. Let the TCAV scores of each group be $TS_{c_1,m,t}$ and $TS_{c_2,m,t}$. Since our objective is to find concepts that generate a higher TCAV score, concept images that have a higher score is preferred. Note that, unlike in the classical preference optimization setting with a human to rank, RLPO preference comes from the TCAV scores (e.g., $TX_{c_1,t} \succ TX_{c_2,t}$). We call this notion RLPO-XAI in some ablation studies below. If the generative model at time t is not capable of generating concepts that are in an explainable state, $\max(TS_{c_1,m,t}, TS_{c_2,m,t}) \leq \eta$, we then perform preference update on SD's weights (more details in Appendix C.5). Following Low-Rank Adaptation (LoRA) (Hu et al., 2021), we only learn auxiliary weights a and b at each time step, and update the weights as $w_{t+1} \leftarrow w_t + \lambda ab$.

As the deep RL agent progresses over time, the states become more relevant as it approaches explainable states (Fig. 1), thus the same action yields increasing rewards over time. To accommodate this, with reference to the rewards defined in Section 3.3, we introduce a parameter ξ, which starts at 0.1 and incrementally rises up to 1 as the preference threshold, η, is approached. Different actions may result in different explainable states, reflecting various high-level concepts inherent to $f(\cdot)$. Some actions might take longer to reach an explainable state. Also, it is possible for different actions to lead to the same explainable state. As the goal is to optimize all states to achieve a common target, DQN progressively improves action selection to expedite reaching these states. Thus, deep RL becomes relevant as it optimizes over time to choose the actions that are most likely to reach an explainable state more efficiently.

4 EXPERIMENTS

To test the effectiveness of our approach, we tested it across multiple models and several classes. We considered three CNN-based classifiers, ResNet-50 (He et al., 2016), GoogleNet (Szegedy et al., 2015), and InceptionV3 (Szegedy et al., 2016), and two transformer-based classifiers, ViT (Dosovitskiy et al., 2020) and Swin (Liu et al., 2021), pre-trained on ImageNet dataset. Unless said otherwise, only GoogleNet results are shown in the main paper. All other model details and results are provided in Appendix C.1 and D.3), respectively. All metrics are defined in Appendix B.1.

4.1 ABLATION STUDY: SEARCH STRATEGIES (WHY DEEP RL?)

We chose DQN as our RL algorithm because of its ability to effectiveness traverse through discrete action space Mnih et al. (2015) (20 unique seed prompts). We assess the effectiveness of RL by disabling the preference optimization step. As shown in Table 1, on GoogleNet classifier, compared to ϵ-greedy methods, RL setup exhibits a higher entropy, average normalized count (ANC), and inverse coefficient of variance (ICV) (See Appendix B.1 for definitions), indicating RL's ability to take diverse actions that results in diverse concepts.

4.2 ABLATION STUDY: SCORING MECHANISMS (WHY TCAV?)

Another important aspect of our setup is the use of TCAV score, an XAI method, to provide preference feedback and calculate rewards. Alternatively, this XAI scoring feedback can also be replaced with human feedback or LLM-based AI feedback. As an additional experiment, to test the effect of human feedback, we conducted human feedback experiment with eight human subjects who provided live human feedback. Further, to evaluate the LLM-based AI feedback, we made use of GPT-4o. More details on the experiment setup and results are provided in Appendix D.2. As shown in Table 3, we concluded that, even though other feedback techniques can be used, XAI-based feedback is best for generating concepts that are important to model with high speed and low computation cost. Though human and AI (GPT4o) are good at correlating semantics, by only looking at test images and concepts instead of model activations, they are not able to provide model specific explanations. Also, human experiments tend to be expensive.

Table 1: Search strategy ablation. We see that RL, compared to ϵ-greedy search, is the best strategy to efficiently explore the search space with high entropy, high average normalized count (ANC) per action, and high inverse coefficient of variance (ICV).

Method	Entropy (↑)	ANC (↑)	ICV (↑)
RL (Ours)	**2.80**	**0.43**	**2.17**
0.25 Greedy	2.40	0.21	1.04
0.5 Greedy	1.95	0.15	0.59
0.75 Greedy	1.85	0.15	0.56

Table 2: Exploration Gap (EG) and Odds calculated based on the responses for ours and retrieval based method, respectively, from the human survey (Appendix D.6).

	Laymen (n=260)	Expert (n=240)
EG (Retrieval)	6.54%	10.45%
EG (Ours)	91.54%	65.45%
Odds (Retrieval)	14.29	8.57
Odds (Ours)	0.09	0.53

Table 3: Scoring mechanisms ablation. We see that RLPO with Explainable AI feedback (RLPO-XAIF), in this case TCAV, is a better choice than RLPO with human feedback (RLPO-HF) and AI feedback (RLPO-AIF).

Method	Class-based Explanations	Model-specific Explanations	Feedback Cost*	Execution Time (↓)
RLPO-HF	✓	✗	NIL	180 ± 30s
RLPO-AIF	✓	✗	>10 GB	72 ± 1.2s
RLPO-XAIF	✓	✓	<1 GB	56 ± 0.7s

* Feedback cost refers to the storage or memory requirements for feedback processing.

Table 4: Novel concepts: $TS_{c,m}$ (TCAV score), CS (Cosine similarity), ED (Euclidean distance), RCS, and RED (CS and ED with ResNet50 embedding)

Methods	Concepts	$TS_{c,m}$(↑)	CS (↓)	ED (↑)	RCS (↓)	RED (↑)
EAC (Sun et al., 2024)	C	1.0	0.76 ± 0.03	7.21 ± 0.63	0.67 ± 0.14	6.34 ± 2.16
Lens (Fel et al., 2024)	C1	1.0	0.77 ± 0.02	7.17 ± 0.34	0.50 ± 0.18	9.70 ± 3.20
	C2	1.0	0.72 ± 0.04	8.02 ± 0.87	0.42 ± 0.10	10.90 ± 2.80
	C3	1.0	0.69 ± 0.05	8.45 ± 0.96	0.45 ± 0.05	11.03 ± 2.17
CRAFT (Fel et al., 2023)	C1	1.0	0.76 ± 0.04	7.37 ± 0.62	0.57 ± 0.16	8.80 ± 3.20
	C2	1.0	0.72 ± 0.02	8.25 ± 0.39	0.50 ± 1.90	9.90 ± 3.40
	C3	1.0	0.73 ± 0.04	7.98 ± 0.79	0.44 ± 0.07	10.80 ± 1.90
RLPO (Ours)	C1	1.0	0.52 ± 0.04	10.48 ± 0.50	0.04 ± 0.01	16.80 ± 1.40
	C2	1.0	0.49 ± 0.02	10.65 ± 0.20	0.02 ± 0.02	17.20 ± 0.80
	C3	1.0	0.49 ± 0.02	10.74 ± 0.30	0.03 ± 0.01	17.60 ± 4.40

4.3 What kind of concepts can RLPO generate?

Novel concepts. As illustrated in Fig. 4, we observed that the RLPO can generate concepts that a human would not typically think of but leads activations of the DNN to trigger. To validate this hypothesis, we conducted a survey to see if humans can think of these generated concepts as important for the neural network to understand a certain class (Table 2, The Exploration Gap (EG) quantifies the proportion of missed optimal actions, defined as $1 -$ Accuracy, highlighting how humans frequently miss the most optimal actions when presented with generated concepts.). As detailed in Appendix D.6, we presented a random class image followed by two concepts, one generated by our method and another from a previous retrieval based method (Fel et al., 2024; 2023). We presented choices from the concepts with high score from both methods and we discovered that while most participants could recognize retrieval-based concepts, only those with domain-specific knowledge could identify generated concepts. This indicates that most people can only identify concepts from a small subset of what $f(.)$ learns during training. Intuitively, when we retrieve concepts from the test class, they tend to be similar to the test images.

We also compare the generated concepts qualitatively and quantitatively. Fig. 4 shows the diversity of concepts generated by our method and other retrieval based methods. Additionally, we also verify

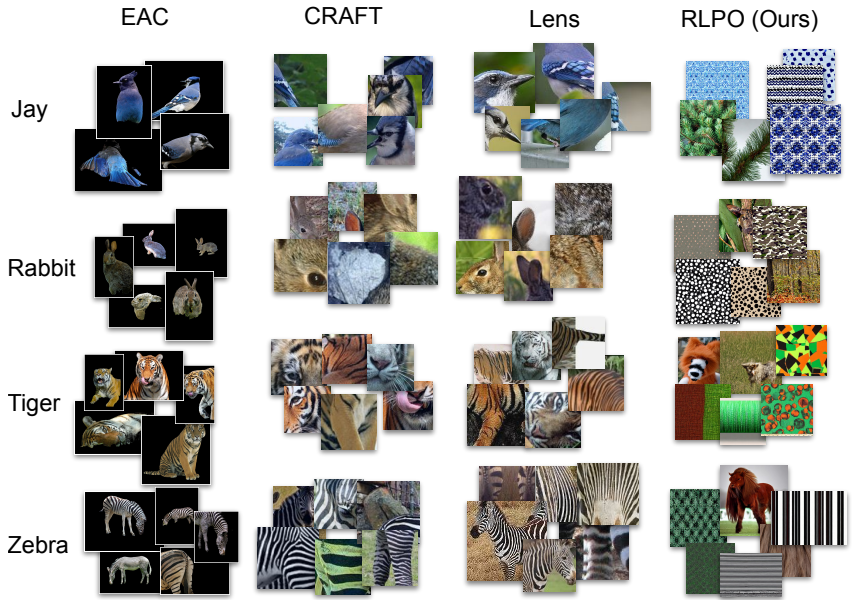

Figure 4: Samples of concepts (with different TCAV scores, not shown here) generated by different methods. Observe that RLPO generates diverse images, not just patches from the test images.

this by computing the vector similarity of the CLIP and ResNET50 embeddings between X_c and X_m for multiple retrieval based methods. As highlighted in Table 4, we observe that retrieval based methods tend to have high cosine similarity between extracted concepts and test images, making them less useful as abstract concepts (e.g., to explain the zebra class, a patch of zebra as a concept is less useful compared to stripes concept).

Abstract concepts. In Fig. 5, we observe the progression of output concepts generated by the SD when RLPO is applied for the seed prompt "zoo," of tiger class. These abstractions hint us about what the model prefers when it is looking for tiger, starting from a four-legged orange furred animal, to black and white stripes with orange furred animal, to black and white stripes with orange furred and whiskers. We obtain concepts with various abstractions by changing η (Currently, it is not possible for our method to decide η to get a particular level of abstraction).

Multiple concepts. Because RLPO algorithm explores various explainable states, we can obtain multiples concepts with varying level of importance. Fig. 6 shows the top three class-level concepts identified by our method for the "zebra" class for the GoogleNet classifier. We see that, each concept set has a different TCAV score associated with them indicating their importance.

4.4 Are the generated concepts reliable?

After generating the concepts, next step is to identify what those concepts signify. To locate where in the class images generated concepts correspond, we made use of CLIPSeg (Lüddecke & Ecker, 2022), a transformer-based segmentation model which takes in concept images as prompts, X_c, and highlights in a test image, $x \in X_m$, which part resembles the input prompt as a heat map. More details on this is available in Appendix D.3.3. As shown in Fig. 6, class image on left highlights the top 3 identified concepts by RLPO. We also compare the output generated by other popular XAI techniques such as LIME and GradCam with ones generated by RLPO. As shown in Fig. 7, we can see that other methods just explains where the model is looking at whereas our approach also explains what type of features is the model focuses on.

After finding the relationship between generated concepts and input images, we need to validate the importance of the identified concepts. To that end, we applied c-deletion, a commonly used validation method in XAI, to the class images for each identified concept. We gradually deleted concept segments based on the segmentation heat map obtained from ClipSeg. The results for the c-deletion are shown in the Fig. 8. We see the area under curve is the highest for the most important

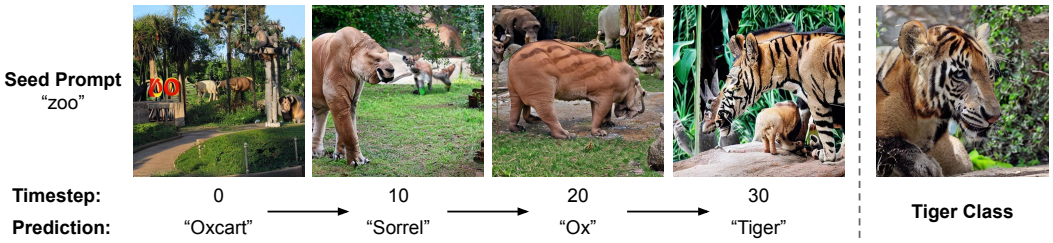

Figure 5: Different levels of abstraction for the "Tiger" class on the GoogleNet classifier are illustrated. The generated image starts as a random "zoo" image and gradually transitions to images with tiger-like features. Observe that the seed prompt "zoo" becomes more animal-like at t=10, gains more stripes at t=20, develops tiger-specific colors at t=30, and progressively refines into a tiger image. The model's prediction also evolves, starting from a random classification of "oxcart" to confidently identifying the generated concept as "tiger".

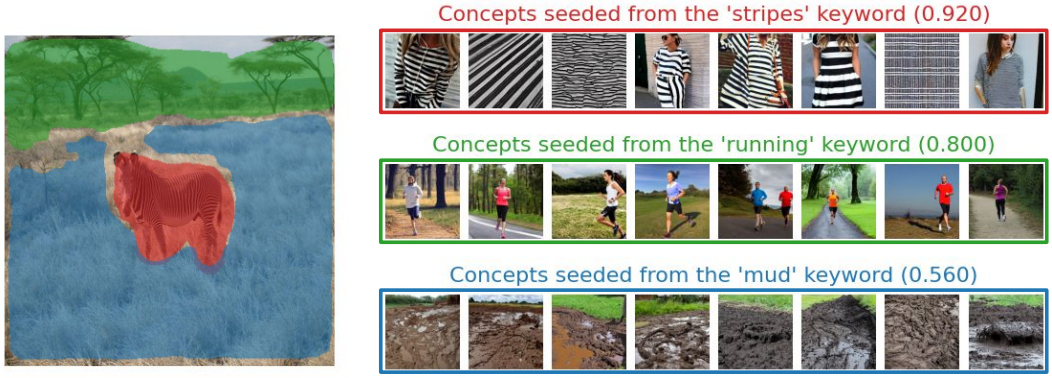

Figure 6: The figure shows the concepts identified by our method and where they are located in the input class image ("zebra" class) for GoogleNet classifier. As highlighted the "stripes" concept image are located near zebra, the "running" concept images, showing trees are highlighting in background of the input image, and the "mud" concept highlighting the grass and soil in the input image. The concepts are ordered in their importance (TCAV score) with "stripes" being the highest and "mud" being the lowest for the selected class.

concept "stripes" and the lowest the least important concept "mud," indicating the order of importance of each concept. More examples on the c-deletion are in Appendix D.3.4.

4.5 Applications and Generalizability

We show how RLPO can be used as a diagnostic tool for the engineers. As a specific application, we see what concepts are removed and added, as well as how the concept importance changes when we fine-tune ResNet50 model on ImageNet to improve accuracy (Fig. 9). More details about the experiment is present in Appendix D.5.

To demonstrate the generalizability of the proposed algorithm, we extended RLPO to generate words in sentiment analysis in NLP. We made use of Mistral-7B Instruct model to generate synonyms of seed prompts and optimized the language model based on preferences from TextCNN model pre-trained on IMDB sentiment dataset. Fig. 10 highlights relevant words with their importance score in the input. More details about the experiment is present in Appendix D.4.

5 Limitations and Conclusions

The process of navigating an infinitely large concept space and generating explainable concepts from textual inputs presents several challenges, particularly when dealing with complex, high-level concepts. We showed how deep RL can guide SD efficiently to navigate this space. However, RLPO also suffers from several limitations. First, this analysis cannot be performed real-time since

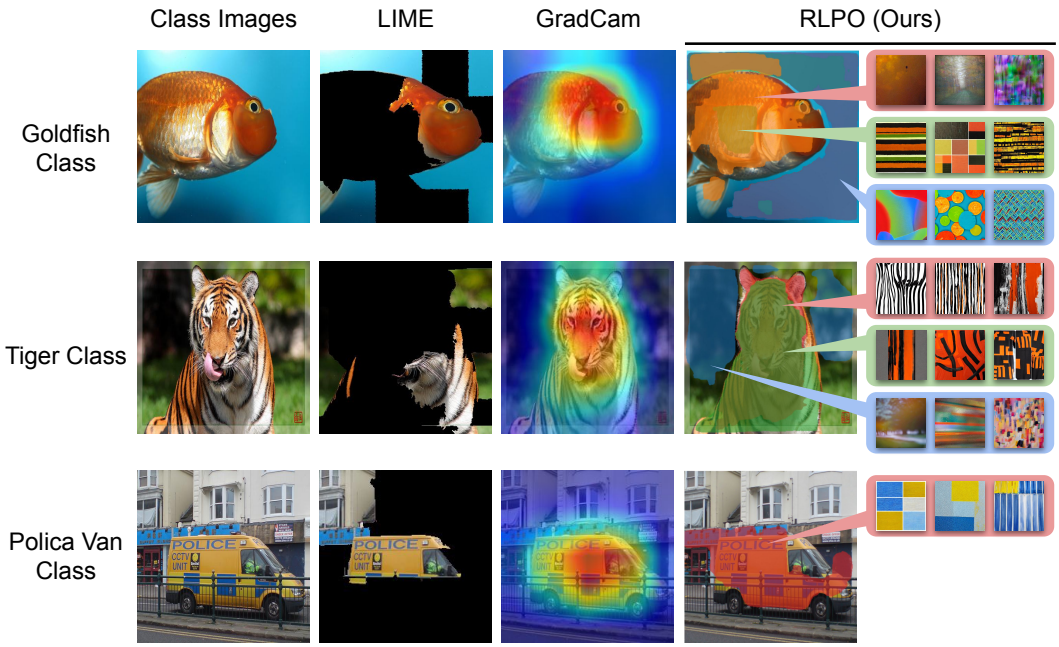

Figure 7: Comparison of concepts identified by different methods. RLPO can show the correspondences between test image and different concepts.

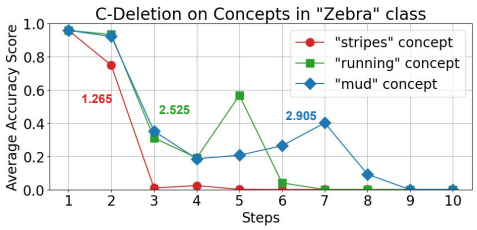

Figure 8: C-deletion. Removing concepts over time to measure the reliability. The colored numbers indicate the area under the curve (the lower the better as it indicates important concepts are removed sooner.)

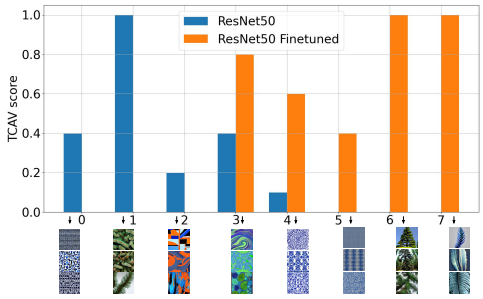

Figure 9: How some eight concepts are shifted from pre (blue) to post (orange) fine-tuning of ResNet50 model on Blue Jay class.

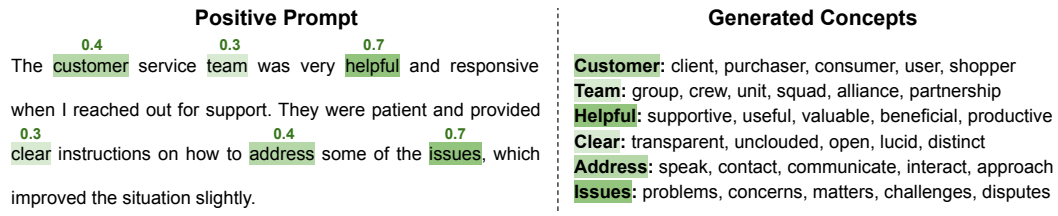

Figure 10: We use sentiment analysis in NLP to show the generalizability of RLPO. Here, concepts tend to be synonyms and the numbers indicate the TCAV scores. Generated concepts explain why a given text is classified as a positive sentiment.

generating images from SD, learning the DQN, and fine-tuning the SD with preferences takes some time. Also, the concepts that our algorithm generates can be diverse as it tries to reveal the concepts inherent to the $f(.)$, making it less domain-specific (e.g., for a medical application, there is a chance it might generate non-medical images if the $f(.)$ activations get excited for non-medical data). As a future extension, we hope to input preferences from both TCAV and application experts while optimizing, making generated explanations even more aligned to specific applications. Despite the challenges, our results show how to leverage the strengths of visual representations and adaptive

learning to provide intuitive and effective solutions for understanding complex, high-level concepts in black-box neural networks.

REFERENCES

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

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*, 30, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Thomas Fel, Agustin Picard, Louis Bethune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2711–2721, 2023.

Thomas Fel, Victor Boutin, Louis Béthune, Rémi Cadène, Mazda Moayeri, Léo Andéol, Mathieu Chalvidal, and Thomas Serre. A holistic approach to unifying automatic concept extraction and concept importance estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in neural information processing systems*, 32, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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. *arXiv preprint arXiv:2106.09685*, 2021.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7086–7096, June 2022.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Lisa Schut, Nenad Tomasev, Tom McGrath, Demis Hassabis, Ulrich Paquet, and Been Kim. Bridging the human-ai knowledge gap: Concept discovery and transfer in alphazero. *arXiv preprint arXiv:2310.16410*, 2023.

Ao Sun, Pingchuan Ma, Yuanyuan Yuan, and Shuai Wang. Explain any concept: Segment anything meets concept-based explanation. *Advances in Neural Information Processing Systems*, 36, 2024.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. *arXiv preprint arXiv:2311.12908*, 2023.

An Yan, Yu Wang, Yiwu Zhong, Chengyu Dong, Zexue He, Yujie Lu, William Yang Wang, Jingbo Shang, and Julian McAuley. Learning concise and descriptive attributes for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3090–3100, 2023.

Yuan Zang, Tian Yun, Hao Tan, Trung Bui, and Chen Sun. Pre-trained vision-language models learn discoverable visual concepts. *arXiv preprint arXiv:2404.12652*, 2024.

APPENDIX

A RELATED WORK

Use of VLM in explaination: Recent advancements in Vision-Language models (VLMs) have open the doors for the use of VLMs in multiple domains, mainly because of their ability to generalize over large amount of data, they can be leveraged to obtain useful information. Work by Sun et al. (2024) present a novel method combining the Segment Anything Model (SAM) with concept-based explanations, called Explain Any Concept (EAC). This method uses SAM for precise instance segmentation to automatically extract concept sets from images, then it employs a lightweight surrogate model to efficiently explain decision made by any neural network based on extracted concepts. Another work by Yan et al. Yan et al. (2023) introduced Learning Concise and Descriptive Attributes (LCDA), which leverages Large Language Models (LLMs) to query a set of attributes describing each class and then use that information with vision-language models to classify images. They highlight in their paper that with a concise set of attributes, they can improve the classifier's performance and also increase interpretability and interactivity for end user.

B DEFINITION, THEOREMS, AND PROVES

Comparative Overlap of Human-Interpretable and Generative Model Concepts in Neural Understanding Tasks ($f(.)$). We formalize this with reference to Fig. 2.

Theorem 1. *Let the set of human-interpretable concepts that the $f(.)$ has learned be $\mathcal{C}_N$, and the concept sets human collected, retrieved though segmentation, and generated using a generative model be $\mathcal{C}_H$, $\mathcal{C}_R$, and $\mathcal{C}_G$, respectively. Then, $|\mathcal{C}_G \cap \mathcal{C}_N| \geq |\mathcal{C}_H \cap \mathcal{C}_N| \geq 0$ and $|\mathcal{C}_G \cap \mathcal{C}_N| \geq |\mathcal{C}_R \cap \mathcal{C}_N| \geq 0$.*

Proof sketch. $\mathcal{C}_H \subseteq \mathcal{C}_G$ and $\mathcal{C}_R \subseteq \mathcal{C}_G \implies |\mathcal{C}_H \cap \mathcal{C}_N| \geq 0$ and $|\mathcal{C}_R \cap \mathcal{C}_N| \geq 0$ □

Proof of Theorem 1.

Definition 3. *Let $\mathcal{C}_H$, $\mathcal{C}_R$, and $\mathcal{C}_G$ denote the sets representing human-interpretable concepts, retrieved concepts, and concepts generated by a generative model, respectively. We define the relationships between these sets as follows:*

$$\mathcal{C}_H \subseteq \mathcal{C}_G \quad \textit{and} \quad \mathcal{C}_R \subseteq \mathcal{C}_G.$$

Property 1. *For any set $\mathcal{C}_i$, where $i \in \{H, R, G\}$, it holds that:*

$$\emptyset \subseteq (\mathcal{C}_i \cap \mathcal{C}_N) \subseteq (\mathcal{C}_i \cup \mathcal{C}_N),$$

where $\mathcal{C}_N$ represents the set of concepts learned by $f(.)$.

For any two sets $\mathcal{A}$ and $\mathcal{B}$, the size of their intersection $|\mathcal{A} \cap \mathcal{B}|$ is non-negative since it represents the number of elements common to both sets. Thus, we have:

$$|\mathcal{C}_H \cap \mathcal{C}_N| \geq 0 \quad \text{and} \quad |\mathcal{C}_R \cap \mathcal{C}_N| \geq 0. \tag{4}$$

Given Definition 3 and Property 1, we assume the following subset relationships between the sets:

$$|\mathcal{C}_H \subseteq \mathcal{C}_G| \quad \text{and} \quad |\mathcal{C}_R \subseteq \mathcal{C}_G|.$$

Case 1. Since $\mathcal{C}_H \subseteq \mathcal{C}_G$, any element $x \in \mathcal{C}_H$ is also in $\mathcal{C}_G$. Therefore, any element $x \in \mathcal{C}_H \cap \mathcal{C}_N$ is also in $\mathcal{C}_G \cap \mathcal{C}_N$. Hence,

$$\mathcal{C}_H \cap \mathcal{C}_N \subseteq \mathcal{C}_G \cap \mathcal{C}_N.$$

Case 2. Similarly, since $\mathcal{C}_R \subseteq \mathcal{C}_G$, any element $x \in \mathcal{C}_R$ is also in $\mathcal{C}_G$. Therefore, any element $x \in \mathcal{C}_R \cap \mathcal{C}_N$ is also in $\mathcal{C}_G \cap \mathcal{C}_N$. Hence,

$$\mathcal{C}_R \cap \mathcal{C}_N \subseteq \mathcal{C}_G \cap \mathcal{C}_N.$$

From Case 1 and Case 2, since $\mathcal{C}_H \cap \mathcal{C}_N$ and $\mathcal{C}_R \cap \mathcal{C}_N$ are subsets of $\mathcal{C}_G \cap \mathcal{C}_N$, it follows that:

$$|\mathcal{C}_H \cap \mathcal{C}_N| \leq |\mathcal{C}_G \cap \mathcal{C}_N| \tag{5}$$

and

$$|\mathcal{C}_R \cap \mathcal{C}_N| \leq |\mathcal{C}_G \cap \mathcal{C}_N| \tag{6}$$

Combining Eqs. 5 and 6 with the non-negativity established in Eq 4, we have:

$$|\mathcal{C}_G \cap \mathcal{C}_N| \geq |\mathcal{C}_H \cap \mathcal{C}_N| \geq 0 \tag{7}$$

and

$$|\mathcal{C}_G \cap \mathcal{C}_N| \geq |\mathcal{C}_R \cap \mathcal{C}_N| \geq 0. \tag{8}$$

□

The Effects of DQN-DPO-based Concept Space Traversal. We now formalize what concepts the DQN has learned, with reference to Fig. 1.

Theorem 2. *When traversing in the concept space, with each reinforcement learning step,*

1. *Case 1: Moving from a proxy state towards an explainable state monotonically increases the reward.*
2. *Case 2: Moving from an explainable state towards the target class does not increase the reward.*

Proof sketch. Obtain the rewards before and after η and compute the difference in reward for each segment. □

Property 2. *As ξ increases, the reward function proportionally amplifies, particularly enhancing the significance of outcomes near t_η, which marks the point beyond which TCAV scores are always 1.*

Proof of Theorem 2. WLOG, let the TCAV score, $S_{c,m,t}$, for concept c and class m at time t be S_t. The reward function is defined as (Section. 3.3),

$$R(t, a) = K \cdot S_t \cdot f(t), \tag{9}$$

for a constant K and a factor,

$$f(t) = \begin{cases} \xi \cdot t & \text{if } t \leq t_\eta, \\ \xi_0 & \text{otherwise,} \end{cases} \tag{10}$$

for positive parameters ξ and ξ_0.

Case 1: Considering the difference in reward function at time t when $t \leq t_\eta$,

$$\begin{aligned} R(t+1, a) - R(t, a) &= K \cdot S_{t+1} \cdot f(t+1) - K \cdot S_t \cdot f(t) \\ &= K \cdot S_{t+1} \cdot \xi \cdot (t+1) - K \cdot S_t \cdot \xi \cdot t \\ &= K \cdot \xi \cdot (S_{t+1} \cdot (t+1) - S_t \cdot t) \\ &= K \cdot \xi \cdot (t(S_{t+1} - S_t) + S_{t+1}) \end{aligned} \tag{11}$$

$$\text{From } (S_{t+1} - S_t) = \frac{h(t+1) - h(t)}{(t+1) - t}) = h'(t), \tag{12}$$

$$R(t+1, a) - R(t, a) = K \cdot \xi \cdot (t \cdot h'(t) + S_{t+1}). \tag{13}$$

Since,

1. S_t is monotonically increasing for $t \leq t_\eta \implies h'(t) > 0$ and
2. $S_t \in [0, 1]$,

$$R(t+1,a) - R(t,a) \geq 0. \tag{14}$$

Case 2: Considering the same difference in rewards for $t \geq t_\eta$.

$$\begin{aligned} R(t+1,a) - R(t,a) &= K \cdot S_{t+1} \cdot f(t+1) - K \cdot S_t \cdot f(t) \\ &= K \cdot S_{t+1} \cdot \xi - K \cdot S_t \cdot \xi_0 \\ &= K \cdot \xi_0 \cdot (S_{t+1} - S_t) \end{aligned} \tag{15}$$

Given that S_t and S_{t_+1} are both outcomes generated from a generative model fine-tuned for a particular concept, $S_{t+1} - S_t \approx 0$ in response to the same action. Hence,

$$R(t+1,a) - R(t,a) \approx 0. \tag{16}$$

□

Theorem 2 characterizes the how TCAV scores (i.e., proportional to rewards) are increased up to η. As a result, as shown in Theorem 3, if the generator moves close to the image class, then the explainer generates images similar to the class. Therefore, by varying η we can generate concepts with different levels of abstractions.

Theorem 3. *As we go closer to the concept class, $|\mathcal{C}_G \cap \mathcal{C}_N|$ becomes larger for generated concepts $\mathcal{C}_G$ and $f(.)$'s internal concepts, $\mathcal{C}_N$.*

Proof sketch. Measure the sensitivity difference between $S_{c_1,m,t}$ and $S_{c_2,m,t}$ as $t \to \infty$. □

Proof of Theorem 3. At each time step t, two sets of samples are generated near $\mathcal{C}_G(t)$ using a generative function $g(.)$, denoted by $s_1(t) = g(\mathcal{C}_G(t))$ and $s_2(t) = g(\mathcal{C}_G(t))$. We define the sensitivity of these samples to the concept class using a measurable attribute, $\sigma(s)$, that quantifies the alignment or closeness of a sample s to the target concept class.

The optimization step at each time step selects the sample with higher sensitivity, denoted by:

$$s_{\text{opt}}(t) = \arg\max\{\sigma(s_1(t)), \sigma(s_2(t))\}$$

The sample with the lower sensitivity is given by

$$s_{\text{min}}(t) = \arg\min\{\sigma(s_1(t)), \sigma(s_2(t))\}$$

Sample $s_{\text{opt}}(t)$ and $s_{\text{min}}(t)$ is then used to adjust $\mathcal{C}_G$, increasing its overall sensitivity to the concept class. Consequently, the sequence of $\mathcal{C}_G$ over time evolves as:

$$\mathcal{C}_G(t+1) = (\mathcal{C}_G(t) \cup \ s_{\text{opt}}(t)) \setminus \ s_{\text{min}}(t)$$

This process incrementally increases the sensitivity of $\mathcal{C}_G(t+1)$ to the concept class, driven by the iterative inclusion of optimized samples.

Given that $\mathcal{C}_N$ is already close to the target concept class, the movement of $\mathcal{C}_G$ through this optimization process indirectly steers $\mathcal{C}_G$ towards $\mathcal{C}_N$. As $\mathcal{C}_G$ evolves in this manner, the overlap between $\mathcal{C}_G$ and $\mathcal{C}_N$ naturally increases, leading to:

$$\lim_{t\to\infty} |\mathcal{C}_G(t) \cap \mathcal{C}_N| \implies \lim_{t\to\infty} \mathcal{C}_G(t) = \mathcal{C}_N.$$

This results from $\mathcal{C}_G(t)$ containing more elements that exhibit higher sensitivity similar to those in $\mathcal{C}_N$, thereby increasing their intersection. □

B.1 Definitions

Entropy: Entropy quantifies the uncertainty or randomness inherent in a probability distribution. For a discrete random variable X with possible outcomes $x_1, x_2, \ldots, x_n$ and corresponding probabilities $P(X = x_i) = p_i$, the entropy $H(X)$ is defined as: $H(X) = -\sum_{i=1}^{n} p_i \log p_i$, where p_i represents the probability of outcome x_i.

Odds: Odds describe how many times an event is expected to happen compared to how many times it is not. They are often used in gambling, sports betting, and statistics. The odds of an event with probability p (where p is the probability of the event happening) are calculated as: $\frac{p}{1-p}$.

Exploration Gap (EG): quantifies the proportion of missed optimal actions, defined as $1 -$ Accuracy, highlighting how humans frequently miss the most optimal actions when presented with generated concepts.

Average Normalized Count (ANC): The ANC is a measure of the central tendency of the normalized action frequencies within a distribution. It provides insight into how the actions are distributed relative to the overall frequency distribution. A high ANC indicates that, on average, the action frequencies are relatively large, meaning that certain actions are more dominant. Conversely, a low ANC suggests that the actions are low and only a few high frequent actions are present. Given by $\frac{1}{n \cdot \max(f)} \sum_{i=1}^{n} f_i$, where f_i is the frequency of action i.

Inverse Coefficient of Variation (ICV): A standardized measure of concentration, calculated as the ratio of the mean to the standard deviation: $\frac{\mu}{\sigma}$. It represents how many standard deviations fit into the mean.

Feedback Cost: Feedback Cost refers to the resource(GPU) expense associated with obtaining feedback during the training of the model.

Execution Time: Execution time refers to the total time taken by a model or algorithm to complete its task from start to finish. This includes the time for data processing, model computation, and generating outputs.

C METHODOLOGY

C.1 MACHINE LEARNING MODELS WE USE

Neural Network Under Test ($f(.)$) : We test RLPO for all the different classification models given below.

1. ResNet50: We utilized a pretrained model from PyTorch torchvision pretrained models with weights initialized from ResNet50_Weights.IMAGENET1K_V2.
2. GoogleNet: We utilized a pretrained model from PyTorch torchvision pretrained models with weights initialized from GoogLeNet_Weights.IMAGENET1K_V1.
3. InceptionV3: We utilized a pretrained model from PyTorch torchvision pretrained models with weights initialized from Inception_V3_Weights.IMAGENET1K_V1.
4. Vision Transformer (ViT): We utilized a pretrained model from PyTorch torchvision pretrained models with weights initialized from ViT_B_16_Weights.IMAGENET1K_V1.
5. Swin Transformer: We utilized a pretrained model from PyTorch torchvision pretrained models with weights initialized from Swin_V2_B_Weights.IMAGENET1K_V1.
6. TextCNN sentiment classification model: We utilized a pretrained model from Captum library. The model was trained on IMBD sentiment dataset.

TCAV logistic model : We utilized a logistic regression model to address classification tasks in TCAV instead of the default SGD (Stochastic Gradient Descent) classifier. This decision was based on our observation that the SGD classifier produced high variance TCAV (Testing with Concept Activation Vectors) scores, which indicated inconsistent model behavior across different runs. We configured the model to perform a maximum of 1000 iterations (max_iter=1000).

Stable Diffusion v1-5 with LoRA : We used our base generation model as SD v1-5 and updated its weights using LoRA during preference optimization step. This version of SD was finetuned from SD v1-2 on "laion-aesthetics v2 5+" dataset with 10% drop in text-conditioning for better CFG sampling. In our experiments, we kept LoRA rank to 8 with a scaling factor of 8 and initial weights were defined from a gaussian distribution. We only targeted the transformer modules of U-Net in the SD architecture.

DQN : We use a DQN with specific parameters tailored to effectively navigate a vast search space. We utilized a small buffer size of 100, which limits the number of past experiences the model can learn from, encouraging more frequent updates. The exploration rate was set at 0.95, prioritizing exploration significantly to ensure thorough coverage of the search space. The batch size was configured to 32. We set the discount factor to 0.99 and the update frequency was set at every four steps. The model updates its parameters with a the soft update coefficient of 1.0. Gradient steps was set to 1 indicating a single learning update from each batch, and gradient clipping was capped at 10 to prevent overly large updates.

BLIP: We utilize the Bootstrapped Language Image Pretraining (BLIP) model for the task of Visual Question Answering (VQA). This model, sourced from the pre-trained version available at 'Salesforce/blip-vqa-capfilt-large', is designed to generate context-aware responses to visual input by leveraging both image and language understanding. The large variant of the BLIP model is fine-tuned for VQA, allowing it to effectively interpret and answer questions based on the visual content provided.

C.2 RLPO ALGORITHM

Algorithm 2 shows the complete algorithm of the algorithm shown in Section 3 algorithm 1.

Algorithm 2 DQN Algorithm with DPO and Adaptive Reward

Input: Set of test images, $f(.)$
Initialize Q-network $Q_\theta(s, a)$ with random weights θ
Initialize replay buffer $\mathcal{D}$ and adaptive parameter $\xi \leftarrow 0.1$
for each episode **do**
  for each time step t **do**
    Observe state s_t and select action a_t based on Q (ϵ-greedy)
    Execute a_t and generate 10 images, divided into two groups G_1 and G_2
    Evaluate TCAV scores $\overline{TCAV}_1$ and $\overline{TCAV}_2$
    if $\max(\overline{TCAV}_1, \overline{TCAV}_2) \leq 0.7$ **then**
      Update policy to favor higher $\overline{TCAV}$ group and perform DPO
      Update $\xi \leftarrow \min(1, \xi + \text{increment})$
    else
      Set $\xi \leftarrow 1$
    end if
    Compute reward $r_t = \xi \cdot \max(\overline{TCAV}_1, \overline{TCAV}_2)$
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample a mini-batch from $\mathcal{D}$
    for each sampled transition (s_i, a_i, r_i, s_{i+1}) **do**
      Compute target $y_i = r_i + \gamma \max_{a'} Q_{\theta'}(s_{i+1}, a')$
    end for
    Compute loss $L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q_\theta(s_i, a_i))^2$
    Perform a gradient descent step to update θ
    Periodically update target network: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$
  end for
end for
Output: Set of concept images

C.3 PREPROCESSING FOR GENERATING THE ACTION SPACE

Steps not discussed in Section 3.2.

Each patch from the test images is passed to the VQA model to extract relevant and useful information about the corresponding class. In this study, we choose BLIP Li et al. (2022) as our VAQ model. We posed a set of targeted questions to the VQA model, aiming to gain insights into the class-specific features represented in the patches. The questions are designed to probe various aspects of the image patches, helping the model focus on class-defining attributes.

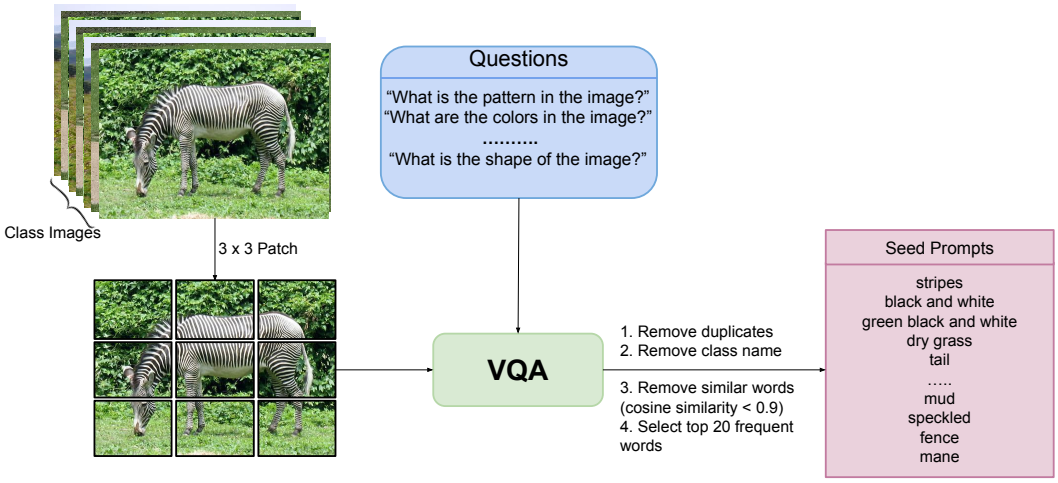

Figure 11: Seed prompt pipeline

1. "What is the pattern in the image?"
2. "What are the colors in the image?"
3. "What is the background color of the image?"
4. "What is in the background of the image?"
5. "What is the primary texture in the image?"
6. "What is the secondary texture in the image?"
7. "What is the shape of the image?"

We then remove stop words and duplicates from the generated responses using lemmantizing and perform a cross-similarity check using CLIP between all the unique words and further filtered words which are more than 95% similar. To further select most relevant keywords to the class images, we perform a VLM check using class images and the extracted keyword to get the softmax score of how much the keyword and image are related. This score is then averaged over all the class images and this average is use to sort the keywords. Now, from the sorted keywords, we select top 20 keywords as our RL action space. The cross-similarity and VLM check are inspired from Zang et al. (2024) where they used a similar filtering setup to remove potentially useless concepts.

C.4 Exploration of Random Gibberish Prompts as Seed Prompts

In this experiment we don't use a VQA to get seed prompts. We choose a random list of incoherent prompts, example shown in Fig. 12. We found that for these prompts it takes a really long time to get some meaningful explanation and in most cases lead to random generation, thus showing importance of starting from proxy concepts.

Gibberish Seed Prompts

1. dKgN MTW8bvbxB6aW1L2TfTuTYZK3He0urbEEmclEpY
2. se-L8fPe19ZzUmuM uDYVYusFnYtNZeFM1YqXdE57Y7OMD3Z80cKwLo5
3. CzKLTlZZnWHjtBn80wIfC z8O
4. mhtxqyH2FBEC
5. SWEC6Wlqfpqaz PQjoGrxIuzm m2ua8oGJySIeG2NqCG9BBvU9Eerj7wheWk7j-t

Figure 12: Sample gibberish seed prompt used with RLPO on GoogleNet classifier to generate concepts.

C.5 PREFERENCE OPTIMIZATION UPDATE FOR STATE SPACE

Steps not discussed in Section 3.4.

The candidate concepts serve as the initial states for the RL agent. From these initial states, the agent takes actions $a \in$ Keywords that leads to multiple subsequent possible states using $g(.)$. These states are then grouped, and the group's sensitivity is compared against Inputs of $f(.)$ using TCAV scores. A higher TCAV score suggests higher sensitivity, indicating that the group is more aligned with $f(.)$'s inputs.

We employ preference optimization over the grouped states to guide states towards explainable concepts. To prevent the model from skipping over explainable states and directly reaching the input domain, we introduce a threshold that limits the application of preference optimization at each step as shown in equation 17.

$$\text{Given two groups of samples } G_1 \text{ and } G_2 \text{ with their average TCAV scores } \overline{TCAV}_1 \text{ and } \overline{TCAV}_2 : \\ \text{if } \max(\overline{TCAV}_1, \overline{TCAV}_2) \leq 0.7, \text{ update } \pi \text{ to favor the group with higher } \overline{TCAV}. \tag{17}$$

To optimize $g(.)$ to find better proxies, for each step in the environment we utilized average TCAV scores $\overline{TCAV}_1$ and $\overline{TCAV}_2$ from G_1 and G_2 to decide between preferred and unpreferred concepts. Lets say $\overline{TCAV}_1 \succ \overline{TCAV}_2$, than we optimize $g(.)$ over the sample S defined as $S = \{(a, x_0^{g1}, x_0^{g2})\}$, where x_0^{g1} and x_0^{g2} are the sample points from the groups on action a. We optimize $g(.)$ using objective 18 to get a new optimzed $g'(.)$ Wallace et al. (2023).

$$\begin{aligned} L(\theta) = &- \mathbb{E}_{(x_0^{g1}, x_0^{g2}) \sim S, t \sim U(0,T), x_t^{g1} \sim q(x_t^{g1} | x_0^{g1}), x_t^{g2} \sim q(x_t^{g2} | x_0^{g2})} \log \sigma \left(-\beta T \omega(\lambda_t) \right. \\ &\left(\|\epsilon^{G1} - \epsilon_{g'(.)}(x_t^{G1}, t)\|_2^2 - \|\epsilon^{G1} - \epsilon_{g(.)}(x_t^{G1}, t)\|_2^2 \right. \\ &\left. \left. - \left(\|\epsilon^{G2} - \epsilon_{g'(.)}(x_t^{G2}, t)\|_2^2 - \|\epsilon^{G2} - \epsilon_{g(.)}(x_t^{G2}, t)\|_2^2 \right) \right) \right) \end{aligned} \tag{18}$$

where $x_t^* = \alpha_t x_0^* + \sigma_t \epsilon^*$, $\epsilon^* \sim \mathcal{N}(0, I)$ is drawn from $q(x_t^* | x_0^*)$. $\lambda_t = \alpha_t^2 / \sigma_t^2$ is the signal-to-noise ratio, and $\omega(\lambda_t)$ is weighting function (constant in practice).

C.6 TCAV SETTING FOR DIFFERENT MODELS

We tested different models on different layers and classes and the summary of our setting across different models is described in table 5.

Table 5: TCAV setting across different models

Models	Layers	ImageNet Classes
ResNet50	penultimate layer	Jay, Tiger, Rabbit & Zebra
GoogleNet	inception4e layer	Goldfish, Tiger, Zebra & Police Van
InceptionV3	Mixed_7c layer	Goldfish, Tiger, Lionfish & Basketball
Vision Transformer (ViT)	heads layer	Goldfish, Golden Retriever, Tiger & Cab
Swin Transformer	head layer	Goldfish, Jay, Siberian husky & Tiger

D EXPERIMENTS

D.1 COMPUTING RESOURCES

The experiments were conducted on a system equipped with an NVIDIA GeForce RTX 4090 GPU, 24.56 GB of memory, and running CUDA 12.2. The system also featured a 13th Gen Intel Core i9-13900KF CPU with 32 logical CPUs and 24 cores, supported by 64 GB of RAM. This setup is optimized for high-throughput computational tasks but the experiments are compatible with lower-specification systems.

D.2 Human and LLM-based AI Feedback Mechanisms

We test other feedback mechanism in RL by replacing the XAI-TCAV feedback with AI and Human feedback's. Herer we discuss the experiental setup and configuration for both experiements.

AI feedback: GPT-4 is leveraged to evaluate the explanatory power of image sets by focusing on concepts related to a target class, a method that aligns with the growing trend of incorporating AI-driven feedback Bai et al. (2022). This approach involves sending a structured prompt to an LLM, asking it to score how well two sets of images explain a target class using a specified concept. The process involves the following.

1. Image Encoding: The images from two sets (concept1 and concept2) are first converted into a base64 format to ensure they can be transmitted via the request as encoded strings.
2. Structured Prompt: A detailed and specific prompt is crafted for the LLM. It asks the model to assess the quality of explanation each image set provides for a particular class through the lens of a specific concept. The prompt used is "Please evaluate each of the following sets of images for how well they explain the class {class_name} via the concept {concept_name}. For each set, provide a numerical score between 0 and 1 (to two decimal places)" The prompt clearly defines how the model should respond, asking for a numerical score between 0 and 1, where:
   (a) 0 indicates that the image set does not explain the class at all via the concept.
   (b) 1 indicates that the image set perfectly explains the class via the concept.
3. LLM-Based Scoring: Once the prompt is sent to the LLM, it evaluates the image sets and provides scores based on its learned knowledge and understanding. The response is parsed to extract the scores for each set of images.

Human feedback: In this experiment, 7 computer science majors provided live feedback after each step of a reinforcement learning process, leveraging their prior knowledge of reinforcement learning with human feedback (RLHF) mechanisms Christiano et al. (2017). The feedback from all participants was averaged to serve as the reward for each step in the RL process. Given the abstract nature of the initial concepts, participants needed to take time to thoughtfully assess each step, which contributed to a lengthier feedback cycle.

D.3 Additional results and analysis

To validate our method for its ability to generate concepts, we tested it with different models and classes. We started it on traditional models, GoogleNet and InceptionV3, and then extended it to transformer-based models, Vision Transformer (ViT) and Swin Transformer, pre-trained on ILSRVC2012 data set (ImageNet) Krizhevsky et al. (2017). We show additional plot in various classes shown in Fig 13,14,15,16,17.

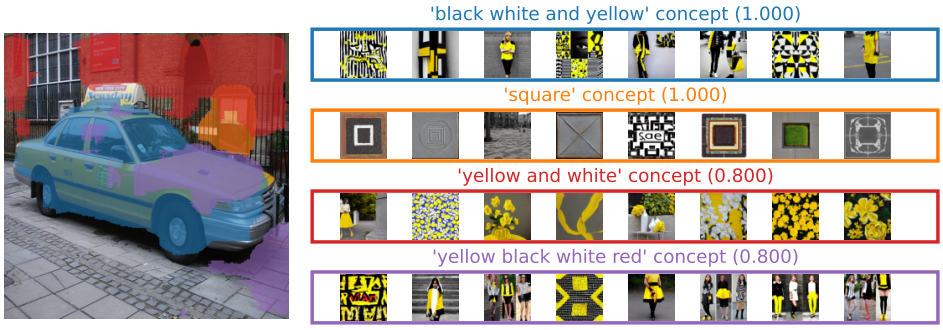

Figure 13: Explanation plot of Cab classification by ViT from RLPO.

Figure 14: Explanation plot of Basketball classification by InceptionV3 from RLPO.

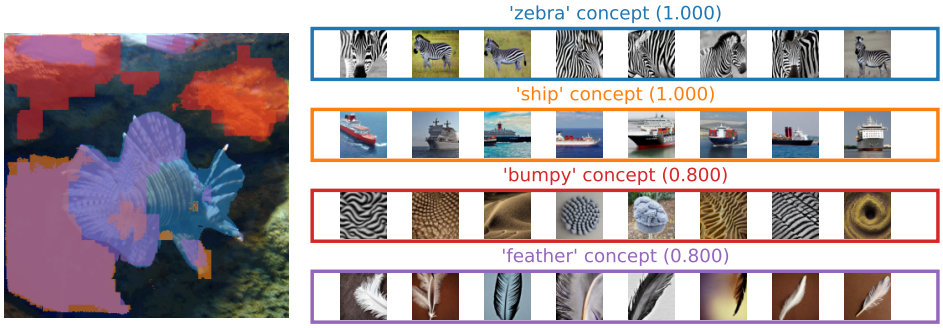

Figure 15: Explanation plot of Lionfish classification by InceptionV3 from RLPO.

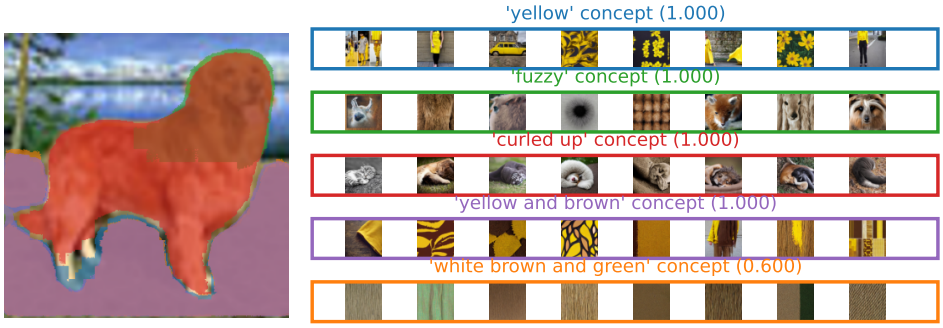

Figure 16: Explanation plot of Golden Retriever classification by ViT from RLPO.

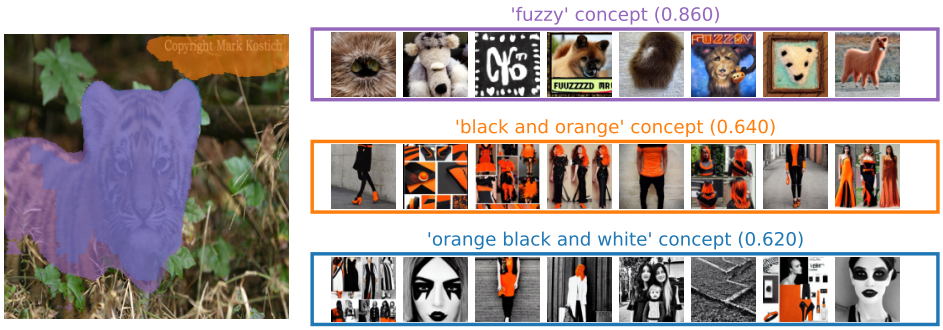

Figure 17: Explanation plot of Tiger classification by GoogleNet from RLPO.

D.3.1 CUMULATIVE REWARDS

The cumulative rewards during training for GoogleNet and InceptionV3 is shown in Fig. 18. For ViT and Swin Transformer it is shown in Fig. 19. This figure illustrates the steady accumulation of rewards over time as they interact with the reinforcement learning environment. All models demonstrate a steady increase in cumulative rewards, the classes with higher reward peak reaches its explinable state faster.

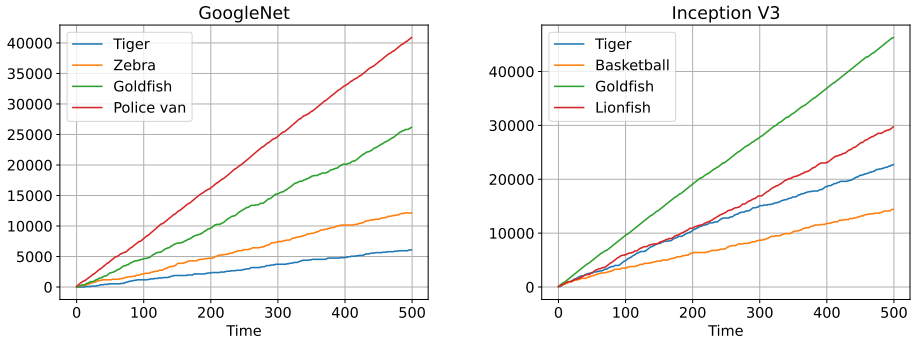

Figure 18: Cumulative rewards on traditional models.

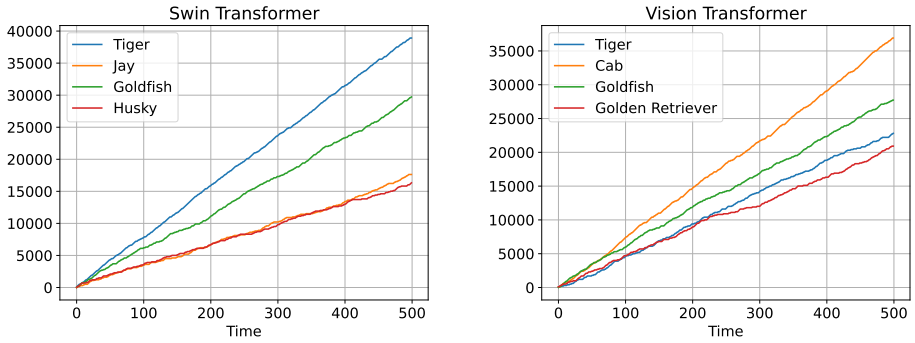

Figure 19: Cumulative rewards on transformer models.

D.3.2 ACTION SELECTION OPTIMIZATION DURING RLPO TRAINING

As shown in Fig. 20, during training with multiple combinations of seed prompts, we observe that the RL agent initially explores various action combinations. However, as training progresses, individual actions become more optimized due to preference optimization (PO). This leads the agent to prefer fewer action combinations, since just choosing one or two actions makes the agent reach an explainable state.

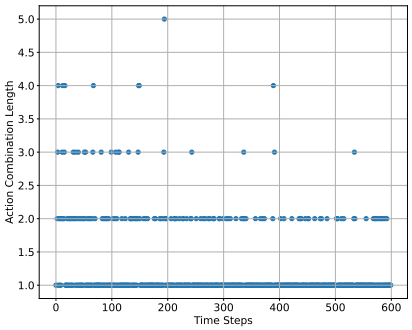

Figure 20: Combined actions (multiple keywords) count over training time

D.3.3 CONCEPT HEATMAP

To determine the relationship between generated concepts and test images, we made use of CLIPSeg transformer model (Lüddecke & Ecker, 2022). We passed generated concepts as visual prompts and test images as query images into the model and it returns a pixel-level heatmap of the probability of visual prompt in the query image. Fig. 21, 22 showcases some examples on concept heatmap indicating the presence of the concept in the image.

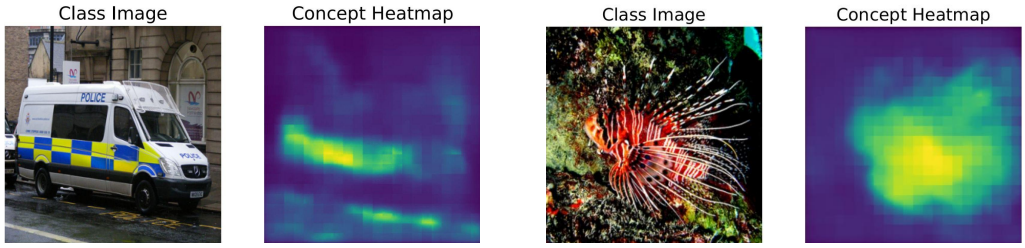

Figure 21: Van class with “white blue and yellow”, Lion fish class with “zebra” seed prompt.

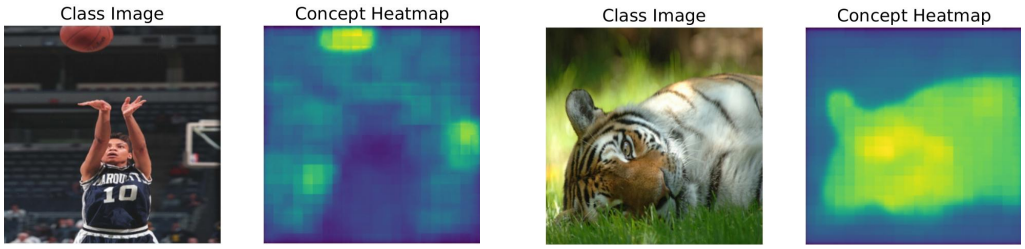

Figure 22: Basketball class with “basket” seed prompt, Tiger class with “orange black and white” seed prompt.

D.3.4 C-DELETION

The central idea behind c-deletion in explainability is to identify and remove parts of the input context that are not crucial for the decision-making process, allowing for clearer insights into how the model arrives at its predictions or actions.

Figure 23: The figure shows c-deletion taking place for different images from “tiger” class over time for “orange black and white” seed concept.

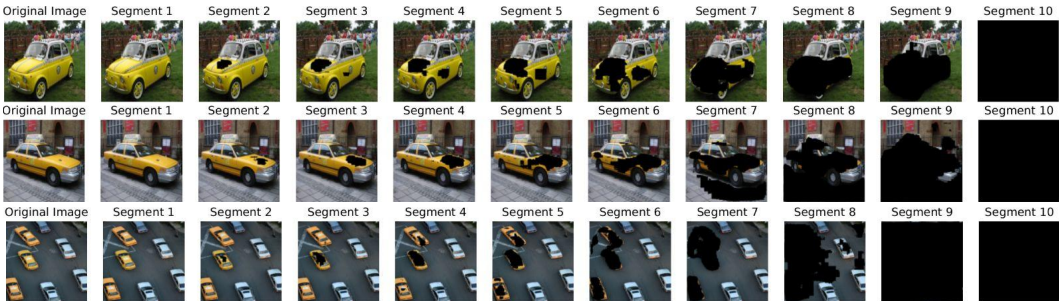

Figure 24: The figure shows c-deletion taking place for different images from “cab” class over time for “yellow and white” seed concept.

C-deletion evaluations assesses the impact of removing certain contextual inputs (features, variables, or states) on a model’s performance as shown in Fig. 25.

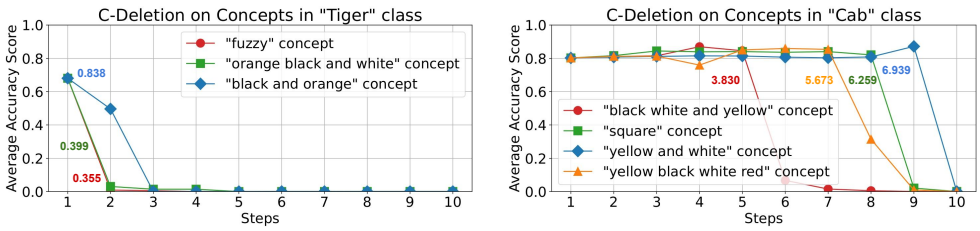

Figure 25: C-deletion. Removing concepts over time to measure the reliability. The colored numbers indicate the area under the curve (the lower the better).

D.4 RLPO IN SENTIMENT ANALYSIS

We extended our method to the NLP domain, successfully identifying which parts of the input contribute to specific outputs. For sentiment analysis, a binary classification problem, we present results for both positive and negative classes.

A list of positive and negative prompts was created, analogous to class images in traditional image classification tasks. Random prompts, similar to those in Fig. 12, were used to simulate random classes. Every word in the prompt, excluding stop words, along with its synonyms, was treated as a concept for this experiment. Synonyms were generated using the Mistral-7B Instruct model, serving a role comparable to the image generation model in image-based settings. We observed that multiple words were identified, along with their influence on the overall prompt, for both classes as shown in Fig. 10 and Fig. 26.

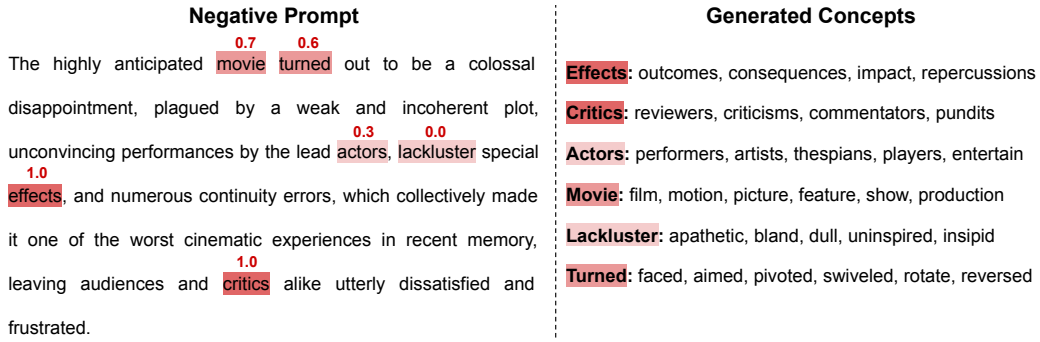

Figure 26: Generated concepts explain why a given text is classified as a negetive sentiment.

D.5 EFFECTS OF FINE-TUNING

When fine-tuning a model, the optimization process updates its weights through gradient-based methods, causing shifts in the concepts (Fig. 9) it learns. These weight adjustments modify how the model attends to different regions or patterns in an image, leading to changes in the internal activation maps and the conceptual understanding of the input. As the model learns new concepts or refines existing ones, it adjusts its feature extraction and decision-making processes to better align with the specific objectives of the fine-tuning task, thereby altering the way it interprets and generates outputs.

To demonstrate this experiment, a ResNet50 model, defined in Appendix C.1, was used. We fine-tuned only the final layer of the neural network by setting the learning rate to 0.001 and momentum to 0.9 using the Stochastic Gradient Descent (SGD) optimizer. Maintaining a low learning rate was crucial to preserving high accuracy. RLPO was then applied to the fine-tuned ResNet50, and a shift in the learned concepts was observed, as shown in Fig. 9. This process highlights the model's sensitivity to fine-tuning and how training on a subset can shift its conceptual interpretation of images.

D.6 HUMAN SURVEY

The survey involved 50 participants, each of whom was shown 10 class images along with two concept options as shown in Fig. 27: one derived from a retrieval-based method and the other generated using RLPO. The participants were divided into Laymen and Experts.

1. Expert: Computer science graduates who are familiar with the concept of explainability and have a working knowledge of AI or machine learning systems.
2. Laymen: Individuals without expertise in computer science, AI, or explainability, representing the general public's perspective.

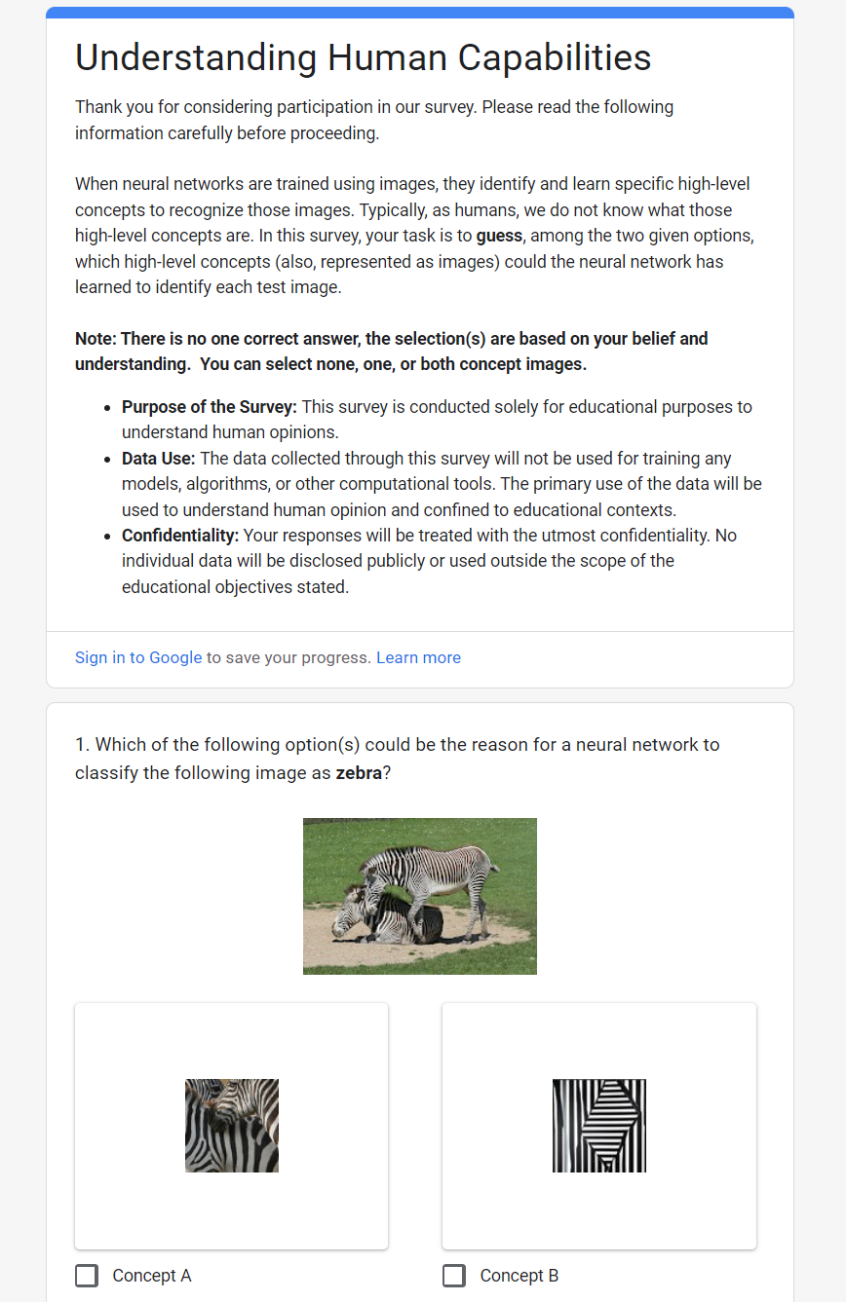

Figure 27: A screenshot from our human survey with instructions and a sample question.