Explainable Concept Generation through Vision-Language Preference Learning for Understanding Neural Networks' Internal Representations

Aditya Taparia
Arizona State University
Email: ataparia@asu.edu

Som Sagar
Arizona State University
Email: ssagar6@asu.edu

Ransalu Senanayake
Arizona State University
Email: ransalu@asu.edu

***Abstract*—Understanding the inner representation of a neural network helps users improve models. Concept-based methods 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 manually collect multiple candidate concept image sets, making the process labor-intensive and prone to overlooking important concepts. Addressing this limitation, in this paper, we frame concept image set creation as an image generation problem. However, since naively using a standard generative model does not result in meaningful concepts, we devise a reinforcement learning-based preference optimization (RLPO) algorithm that fine-tunes a vision-language generative model from approximate textual descriptions of concepts. Through experiments, we demonstrate our method's ability to efficiently and reliably articulate diverse concepts that are otherwise challenging to craft manually, enabling interpretable human oversight in settings such as robotics. Github: https://github.com/aditya-taparia/RLPO**

I. 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. 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) [7], have gained great popularity. 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.

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.

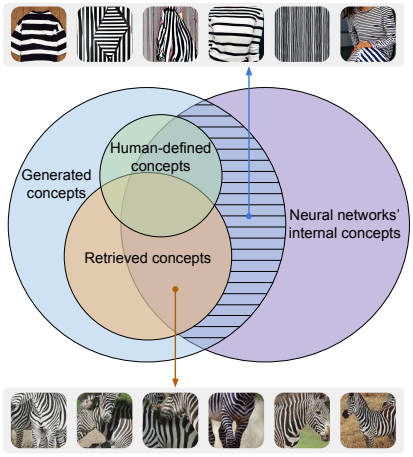

Fig. 1. Humans can imagine (green) a few concepts to understand neural networks' representations (purple). Some other concepts can be retrieved from test images themselves through, for instance, segmentation (orange). However, if we generate concepts (blue), they will capture even a broader set of concepts. RLPO is designed to make this generation process more targeted toward neural networks' representations (purple $\cap$ blue).

As attempts to automatically discover and create such concept sets, several work has focused on segmenting the image and use them as potential concepts, either directly [5] or through factor analysis [3, 4]. In such methods, which we call as retrieval methods, because the extracted concept set is already part of the test images, it is difficult for them to imagine new concepts that do not have a direct pixel-level resemblance to the original image class. For instance, as shown in Fig. 1, it is arguable if some patches of zebra—instead of stripes—qualify

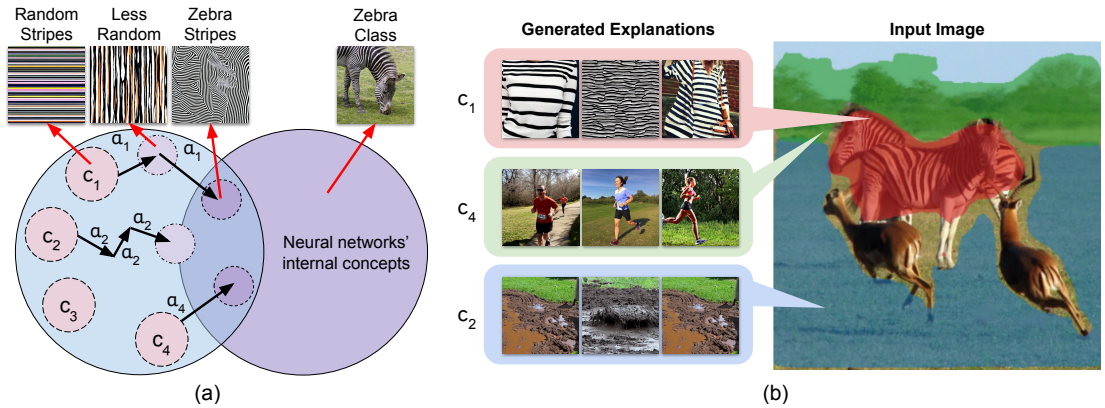

Fig. 2. (a) Our proposed algorithm, RLPO, iteratively refines the concepts c_i that are generated by a Stable Diffusion (SD) model by optimizing SD weights based on an action a_i. Each step in this update process provides an explanation at a different level of abstraction. (b) Three concepts identified by our approach for the zebra class. Concepts are represented as images generated by SD.

as high-level concepts to explain the zebra class.

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 produce noise-free, realistic images. Nevertheless, since a generative model can generate arbitrary images, we need to guide it to produce what we desire by using text prompts. One obvious approach is to engineer long, descriptive text prompts to generate concepts. However, engineering such prompts is not realistic. Therefore, to automate prompting, we extract keywords related to the image using an image-to-text model (we call them seed prompts). As shown in Fig. 2, we propose a reinforcement learning-based preference optimization (RLPO) algorithm that guides the generative model to automatically generate meaningful concepts based on these seed prompts. Preferences are solely decided by the explanation score—not by a human—that the deep RL (DRL) algorithm is trying to optimize.

II. 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 [7]. 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). 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 scores (TS) 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,

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

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, $\{f_1(x); x \in X_r\}$.

ACE [5] introduced a way to automatically find relevant concepts by extracting them from the input class. It uses image segmentation of multiple sizes 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 very noisy. EAC [17] extracts concepts through segmentation. CRAFT [3] introduced a recursive strategy to detect and decompose concepts across layers. Lens [4] 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. In contrast, we generate concepts from a generative model. Under generative models, LCDA [22] simply queries an LLM to get attributes but does not generate concepts.

Deep Q Networks (DQN): DQN [12] is a DRL 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 $argmaxQ(s_{next}, a)$ which is a copy of the Q-network with parameters θ', is updated less frequently to provide stable targets for Q-value updates,

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

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. DQNs are used for controlling robots [20, 16, 1], detecting failures [14], etc.

Preference Optimization: Optimizing generative models directly through preference data was first proposed in Direct Preference Optimization (DPO) [13]. 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 [21], 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.

III. Methodology: Reinforcement Learning-based Preference Optimization

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 relevant to the neural networks' decision-making process. We leverage 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, RLPO, uses RL to pick potential keywords from a predefined list and iteratively optimizes stable diffusion weights to generate images that have a preference for higher TCAV scores. This process is described below.

Algorithm 1: The RLPO algorithm. Appendix B for the expanded algorithm.

Input : Set of test images $f(\cdot)$
Run pre-processing and obtain 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 and G_2;
    Evaluate TCAV scores TS_1 and TS_2;
    Update SD based on the better score;
    Compute reward;
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. The core search algorithm that we train is a DQN.

A. 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.

B. Deep Reinforcement Learning Formulation

Our objective of using deep RL is automatically controlling text input of Stable Diffusion. As text input, we start with $\mathcal{K}$ seed prompts from Section III-A, 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: It 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 $\xi_{t+1,k} \leftarrow \min\left(1, \frac{\xi_{t,k}+1}{T}\right)$, where T is total number of RL steps. Therefore, the expected cumulative adjusted reward is $R(\pi) = \mathbb{E}\Big[\sum_{t=0}^{T} \xi_t \cdot r_t(s_t, a_t)\Big]$.

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,

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

C. 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., $TX_{c_1,t} \succ TX_{c_2,t}$). If the generative model at time t is not capable of generating concepts that are in an explainable state,

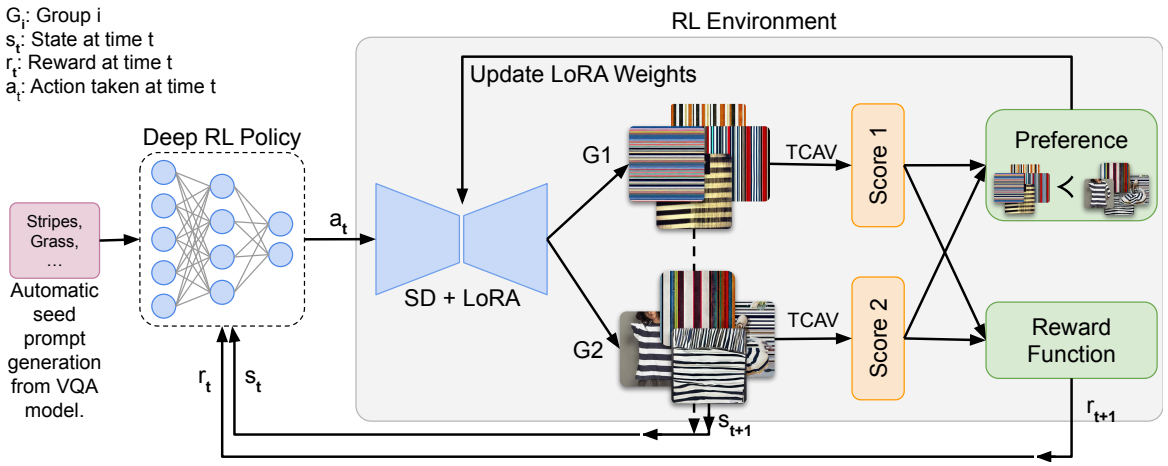

Fig. 3. Overview of the RLPO framework with its dynamic environment interaction. The RL policy selects actions (seed prompts) which generates concept sets (G1, G2) scored through TCAV. Reward is calculated based on the scores obtained for both the sets. Simultaneously, best set is determined based on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

$\max(TS_{c_1,m,t}, TS_{c_2,m,t}) \leq \eta$, we then perform preference update on SD's weights. Following Low-Rank Adaptation (LoRA) [6]—a method that allows quick SD adaptation with a few samples,— 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, thus the same action yields increasing rewards over time. To accommodate this, with reference to the rewards defined in Section III-B, 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. 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.

IV. Experiments

To verify the effectiveness of our approach, we tested it across multiple models and several classes. We considered two CNN-based classifiers, GoogleNet [18] and InceptionV3 [19], and two transformer-based classifiers, ViT [2] and Swin [10], 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 H.

TABLE I
Exploration Gap (EG) and Odds for different methods based on the human survey (Appendix I). This verifies that RLPO can generate concepts that human cannot think of.

	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

A. Concepts generated by RLPO

Objective of RLPO is to automatically generate concepts that a human or a retrieval method cannot propose but the neural network has indeed learned (i.e., gets activated). We observed that the RLPO can generate diverse set of 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 DNN to understand a certain class.

As detailed in Appendix I, during our human survey, we presented a random class image followed by two concepts, one generated by our method and another from a previous retrieval-based method [4, 3]. While the choices were similar in terms of XAI-score from both methods, we discovered that most participants recognize retrieval-based concepts, and only those with domain-specific knowledge could identify generated concepts as important. As highlighted in Table I, high Exploration Gap (EG) of our method 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 further verify that the generated concepts have the following properties.

Diverse representations per concept. We verify the diversity (e.g., different types of stripes) of generated and retrieval-based concepts by computing the vector similarity of the CLIP and ResNET50 embeddings between X_c and X_m for different classes. As highlighted in Table II, we observe that concepts from retrieval-based methods tend to have high cosine similarity with 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).

Multiple concepts per class. Since RLPO algorithm explores various explainable states, we can obtain multiples concepts (e.g., stripes, savanna) with varying level of importance. Fig. 4 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

TABLE II

NOVEL CONCEPTS: $TS_{c,m}$ (TCAV SCORE), CS (COSINE SIMILARITY), ED (EUCLIDEAN DISTANCE), RCS, AND RED (CS AND ED WITH RESNET50 EMBEDDING). THIS INDICATES THAT RLPO CAN GENERATE A DIVERSE SET OF CONCEPTS THAT TRIGGERS THE NETWORK.

Methods	Concepts	$TS_{c,m}$(↑)	CS (↓)	ED (↑)	RCS (↓)	RED (↑)
EAC [17]	C	1.0	0.76 ± 0.03	7.21 ± 0.63	0.67 ± 0.14	6.34 ± 2.16
Lens [4]	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 [3]	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

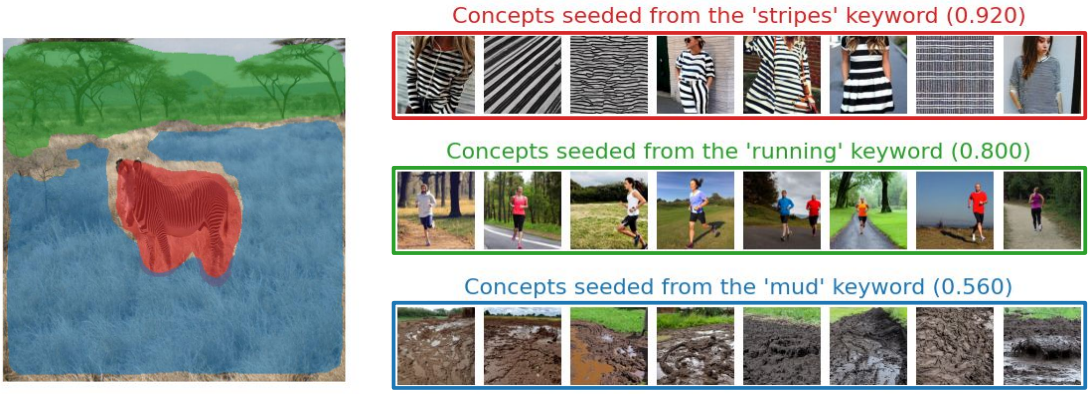

Fig. 4. The figure shows the concepts generated by our method and where they are located in the input image ("zebra" class) for GoogleNet classifier. As highlighted the "stripes" concept images are located near zebra, the "running" concept images, showing trees, highlight the background, and the "mud" concept highlights 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.

with them indicating their importance.

B. Are generated concepts correct?

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 [11], 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 H3. As shown in Fig. 4, 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.

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 heat map obtained from ClipSeg. The results for the c-deletion are shown in the Fig. 5. We see the area under curve is the highest for the most important concept "stripes" and the lowest for least important concept "mud," indicating the order of importance of each concept.

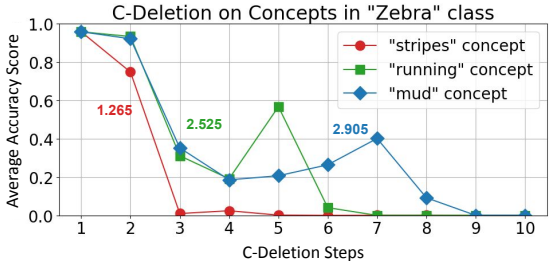

Fig. 5. C-deletion. Removing concepts over time to measure the reliability. The colored numbers indicate the area under the curve.

More examples on the c-deletion are in Appendix H4.

C. How are generated concepts useful to engineers?

To verify the usability of the generated concepts, we conducted a human study with 19 ML engineers. We first provided them the concept generated by our method for "zebra" class and ask them to choose relevant concepts for GoogleNet to classify a zebra without telling them that all shown images are actual concepts. All the engineers selected the "stripes" concept to be important while some also selected the "mud" concept. But most missed the "running" concept. This indicates that engineers cannot think of all the important concepts that gets the

neural network activated. In the next step, we showed engineers the concept-explanation mapping on a random input image (similar to Fig. 4) and asked them if the provided explanation helped them understand the model better and if it provided new insights. 94.7% of the engineers agreed that the explanation helped in better understanding the neural network and 84.2% agreed that it provided new insights. This result shows that the new concepts discovered by our proposed method help engineers discover new patterns that they did not imagine before (More details in Appendix J).

We now demonstrate how engineers can use these newly revealed information about concepts to improve the model. As identified by RLPO, for Tiger class, the base GoogleNet model gives equal importance to both foreground (highlighted by concepts "orange black and white" and "orange and black") and background (highlighted by concepts "blurry") in the input (see Fig. 8 in Appendix H for example explanations). As shown in Fig. 6, when we fine-tune the GoogleNet on images of Tiger-related concepts, we see that the fine-tuned model now focuses more on tiger than the background while maintaining a similar accuracy (65.6%).

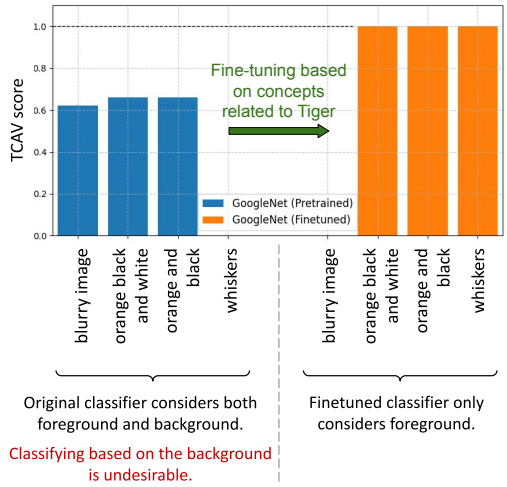

Fig. 6. Usefulness of RLPO. Fine-tuning GoogleNet based on generated concepts for the Tiger class.

V. Limitations and Conclusions

Navigating an infinitely large concept space and generating explainable concepts from textual inputs pose several challenges. We showed how deep RL can guide SD efficiently to navigate this space. However, RLPO also suffers from some limitations. 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 aim to input preferences from both TCAV and domain 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 neural networks. While our experiments focus on visual classifiers, the core idea behind RLPO on identifying meaningful and important concepts learned by a model can be extended to robotic systems. In scenarios where a robot takes unexpected actions, RLPO can offer a way to surface the abstract visual concepts influencing those decisions, allowing human operators to interpret, question, or correct behaviors in the loop. As a future work, we aim to apply RLPO to embodied agents and decision-making pipelines in robotics, making concept-based feedback a practical component in human-in-the-loop robot learning.

References

[1] Harrison Charpentier, Ransalu Senanayake, Mykel Kochenderfer, and Stephan Günnemann. Disentangling epistemic and aleatoric uncertainty in reinforcement learning. In *ICML 2022 Workshop on Distribution-Free Uncertainty Quantification*, 2022.

[2] 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.

[3] 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*, pages 2711–2721, 2023.

[4] 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.

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

[6] 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.

[7] 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*, pages 2668–2677. PMLR, 2018.

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

[9] 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*, pages 12888–12900. PMLR, 2022.

[10] 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*, pages 10012–10022, 2021.

[11] 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)*, pages 7086–7096, June 2022.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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*, pages 1–9, 2015.

[19] 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*, pages 2818–2826, 2016.

[20] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *arXiv preprint arXiv:2408.03539*, 2024.

[21] 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.

[22] 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*, pages 3090–3100, 2023.

[23] 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. The Rationales Behind Design Choices

Rationale 1: Why concept generation is a better idea. If we use concept-based explanation the traditional way [7, 15], then the end users need to manually guess what concepts to test for. Automatically retrieving the concept set by segmenting test images [17] also results in a limited concept set. In contrast, a SOTA generative model can generate high quality images.

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. 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 outcome by using 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 a 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.

B. RLPO algorithm

Algorithm 2 presents the detailed version of the algorithm introduced in Section III, 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 $\xi \leftarrow 1$;
    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 $y_i = r_i + \gamma \max_{a'} Q_{\theta'}(s_{i+1}, a')$;
    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: $\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$;

Output : Set of concept images

C. 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 [12] (20 unique seed prompts). We assess the effectiveness of RL by disabling the preference optimization step. As shown in Table III, on the GoogleNet classifier, compared to ϵ-greedy methods, the RL setup exhibits higher entropy, average normalized count (ANC), and inverse coefficient of variance (ICV), indicating RL's ability to efficiently explore across diverse actions.

TABLE III

SEARCH STRATEGY ABLATION. WE SEE THAT RL, COMPARED TO ϵ-GREEDY SEARCH, IS THE BEST STRATEGY TO EXPLORE THE SEARCH SPACE WITH HIGH ENTROPY, AVERAGE NORMALIZED COUNT (ANC) PER ACTION, AND 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

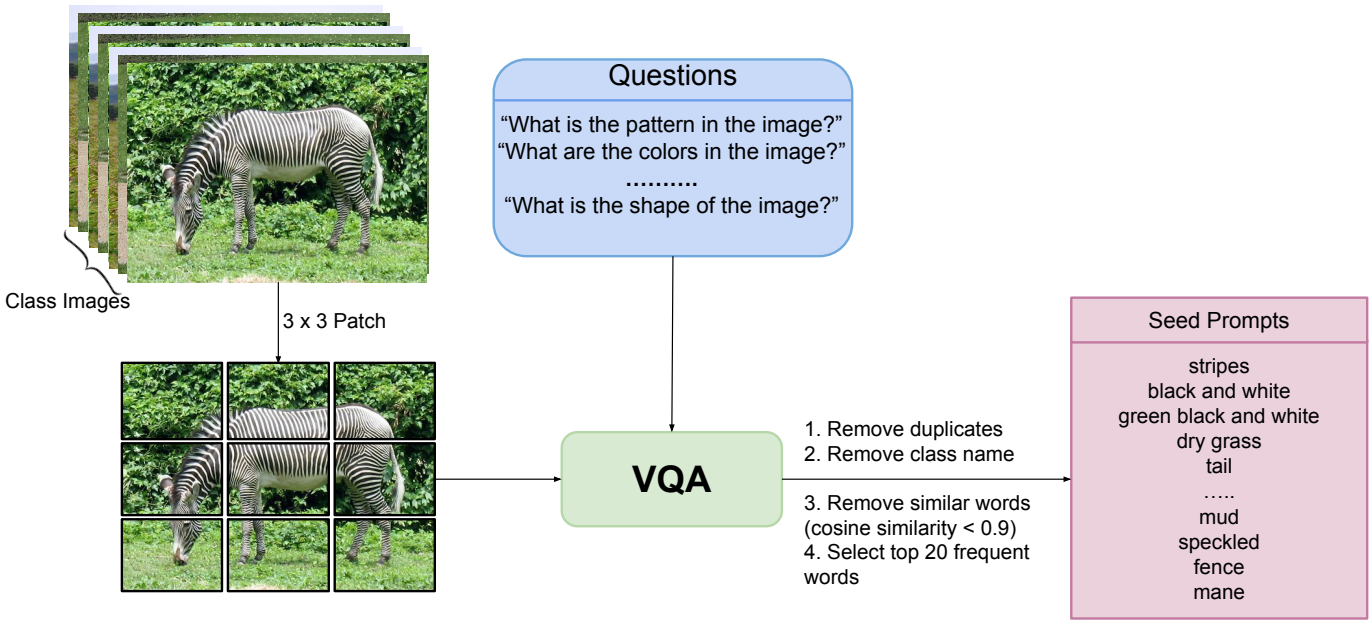

Fig. 7. Seed prompt pipeline

D. Preprocessing for generating the action space

Steps not discussed in Section III-A.

As shown in Fig. 7, 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 [9] 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.

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 [23] where they used a similar filtering setup to remove potentially useless concepts.

E. Preference optimization update for state space

Steps not discussed in Section III-C.

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 4.

$$\begin{aligned} &\text{Given two groups of samples } G_1 \text{ and } G_2 \text{ with their average TCAV scores } \overline{TCAV}_1 \text{ and } \overline{TCAV}_2 : \\ &\text{if } \max(\overline{TCAV}_1, \overline{TCAV}_2) \leq 0.7, \text{ update } \pi \text{ to favor the group with higher } \overline{TCAV}. \end{aligned} \tag{4}$$

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 5 to get a new optimzed $g'(.)$ [21].

$$\begin{aligned} L(\theta) = &- \mathbb{E}_{(x_0^{g1}, x_0^{g2}) \sim S, t \sim U(0,T), x_t^{g1} \sim q(x_t^{g1}|x_0^{g1}), x_t^{g2} \sim q(x_t^{g2}|x_0^{g2})} \log \sigma \left(-\beta T \omega(\lambda_t) \right. \\ &\left(\|\epsilon^{G1} - \epsilon_{g'(.)}(x_t^{G1}, t)\|_2^2 - \|\epsilon^{G1} - \epsilon_{g(.)}(x_t^{G1}, t)\|_2^2 \right. \\ &\left. \left. - \left(\|\epsilon^{G2} - \epsilon_{g'(.)}(x_t^{G2}, t)\|_2^2 - \|\epsilon^{G2} - \epsilon_{g(.)}(x_t^{G2}, t)\|_2^2 \right) \right) \right) \end{aligned} \tag{5}$$

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).

F. 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 IV.

TABLE IV
TCAV SETTING ACROSS DIFFERENT MODELS

Models	Layers	ImageNet Classes
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

G. 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.

H. 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) [8]. We show additional plot in various classes shown in Fig 8,9,10,11,12.

1) Cumulative rewards: The cumulative rewards during training for GoogleNet and InceptionV3 is shown in Fig. 13. For ViT and Swin Transformer it is shown in Fig. 14. 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.

2) Action Selection Optimization During RLPO Training: As shown in Fig. 15, 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.

Fig. 8. Explanation plot of Tiger classification by GoogleNet from RLPO.

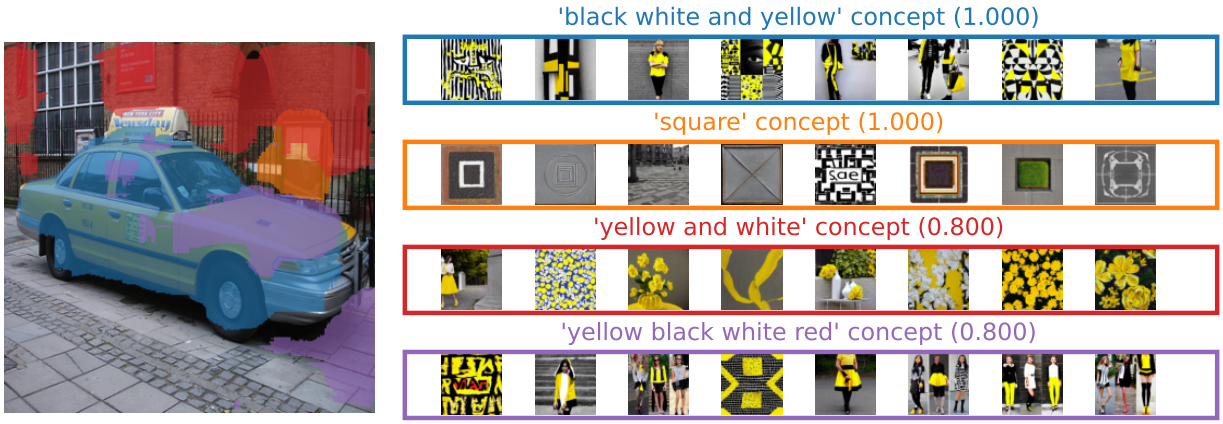

Fig. 9. Explanation plot of Cab classification by ViT from RLPO.

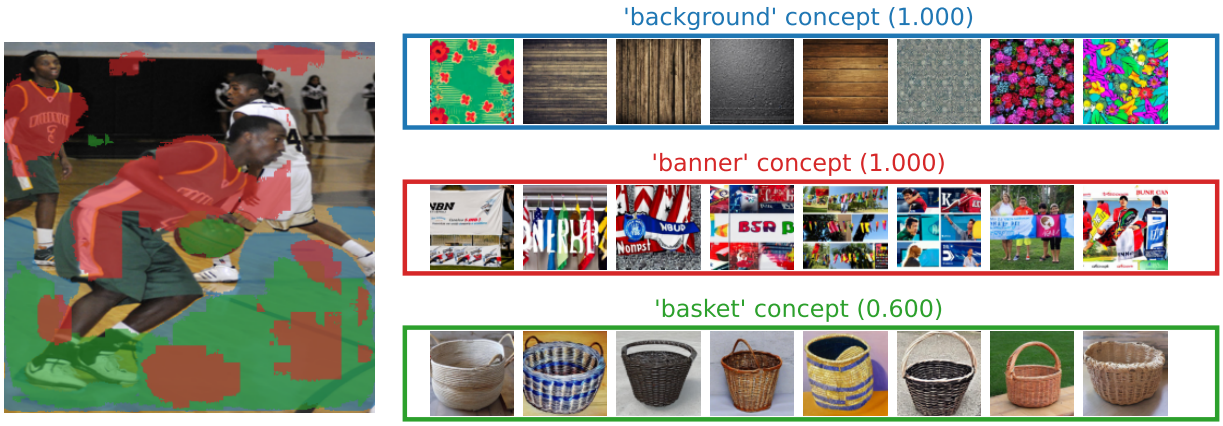

Fig. 10. Explanation plot of Basketball classification by InceptionV3 from RLPO.

3) Concept heatmap: To determine the relationship between generated concepts and test images, we made use of CLIPSeg transformer model [11]. 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. 16, 17 showcases some examples on concept heatmap indicating the presence of the concept in the image.

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

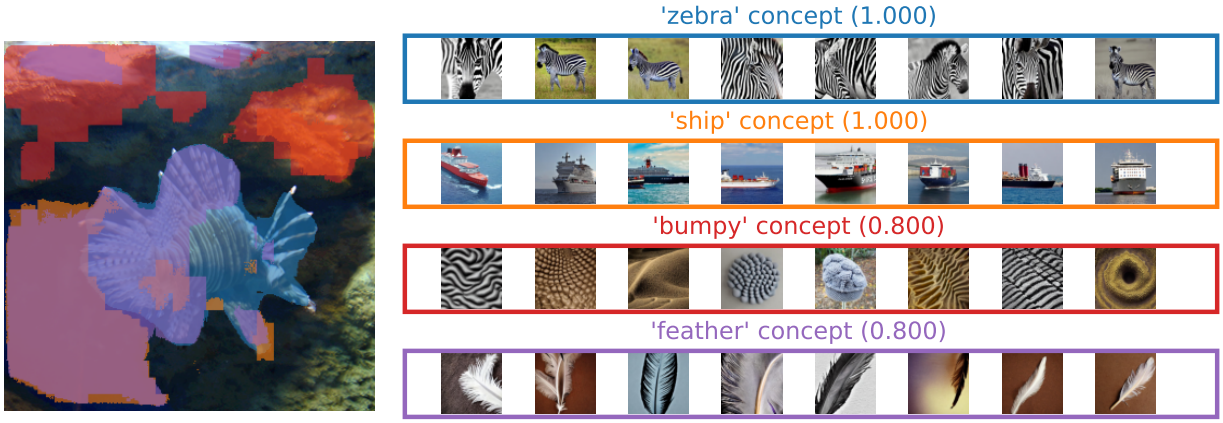

Fig. 11. Explanation plot of Lionfish classification by InceptionV3 from RLPO.

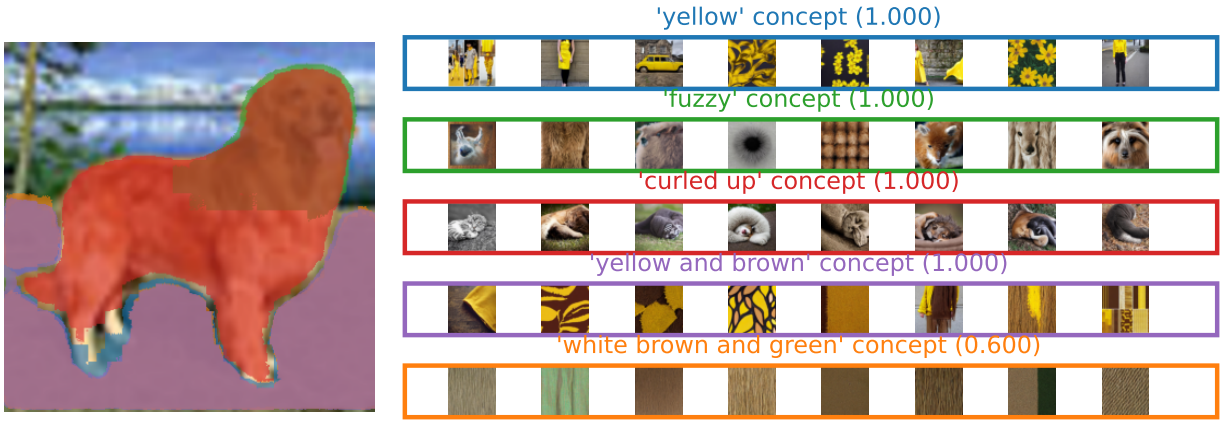

Fig. 12. Explanation plot of Golden Retriever classification by ViT from RLPO.

actions.

C-deletion evaluations assesses the impact of removing certain contextual inputs (features, variables, or states) on a model's performance as shown in Fig. 20.

I. Human survey: Understanding human capabilites

The survey involved 50 participants, each of whom was shown 10 class images along with two concept options as shown in Fig. 21: 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.

J. Human study: Usability

We conducted a human study to measure the usefulness of the provided explanation. The study involved 19 ML engineers. Fig. 22 shows the survey used while conducting human study to measure usability.

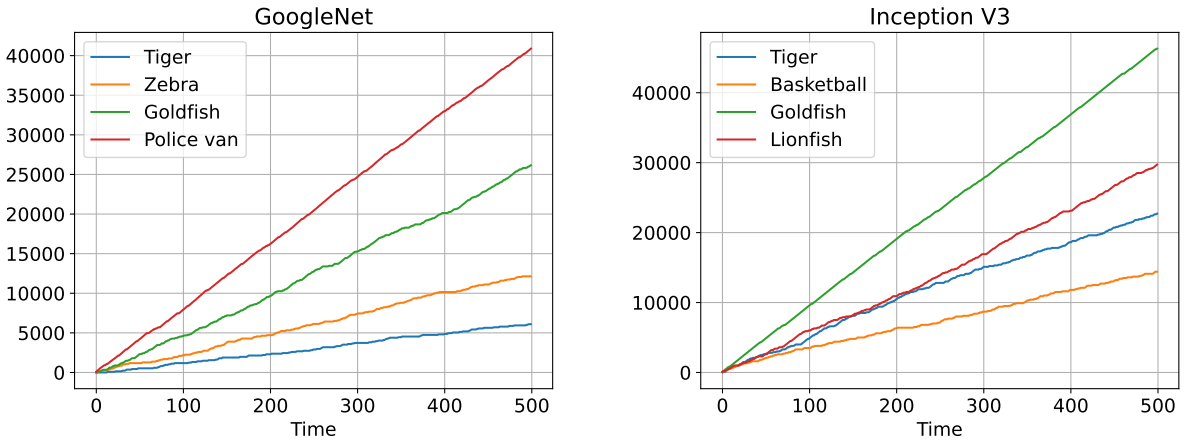

Fig. 13. Cumulative rewards on traditional models.

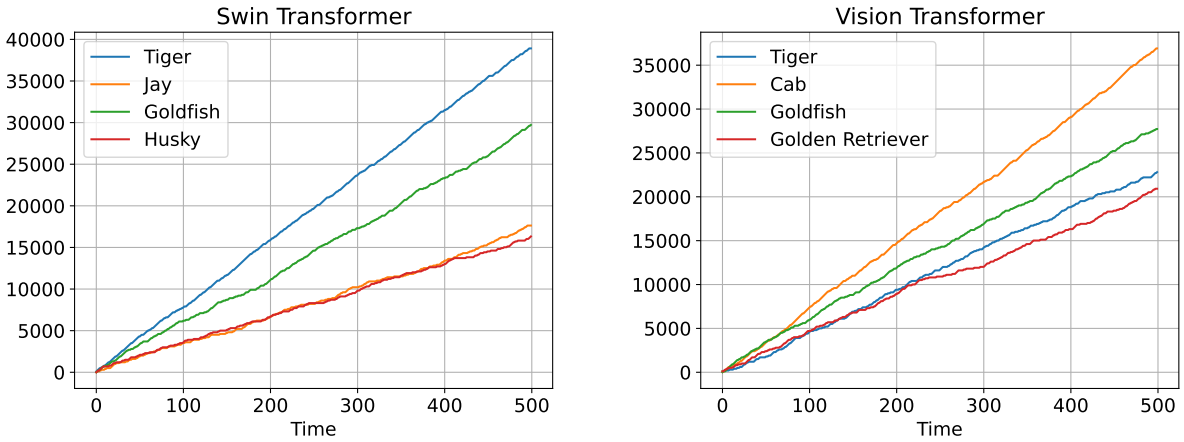

Fig. 14. Cumulative rewards on transformer models.

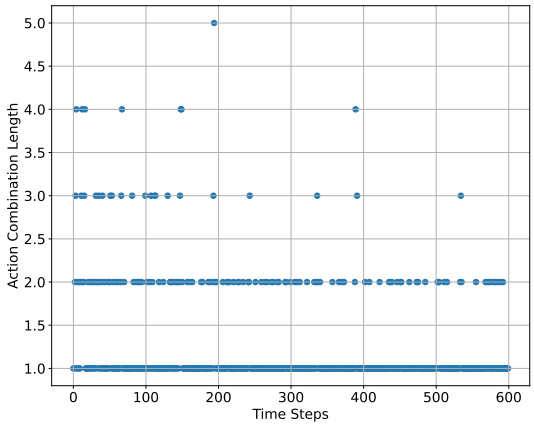

Fig. 15. Combined actions (multiple keywords) count over training time

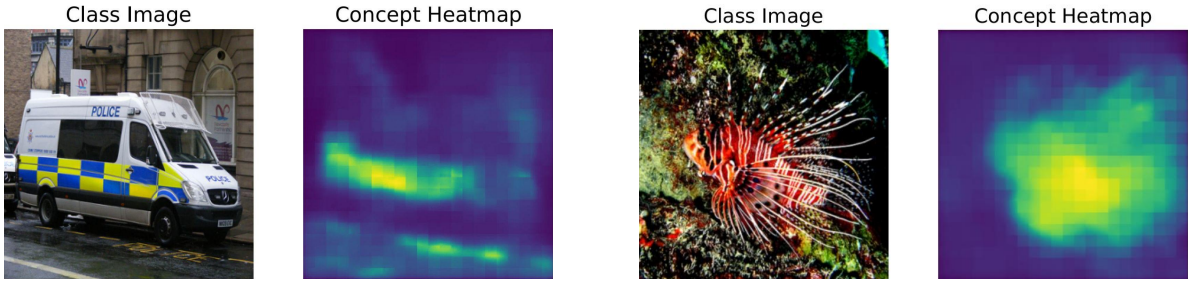

Fig. 16. Van class with “white blue and yellow”, Lion fish class with “zebra” seed prompt.

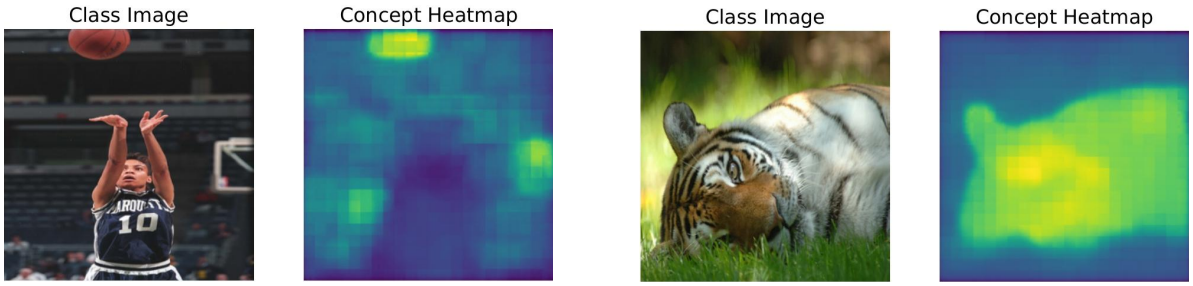

Fig. 17. Basketball class with “basket” seed prompt, Tiger class with “orange black and white” seed prompt.

Fig. 18. The figure shows c-deletion taking place for different images from “tiger” class over time for “orange black and white” seed concept.

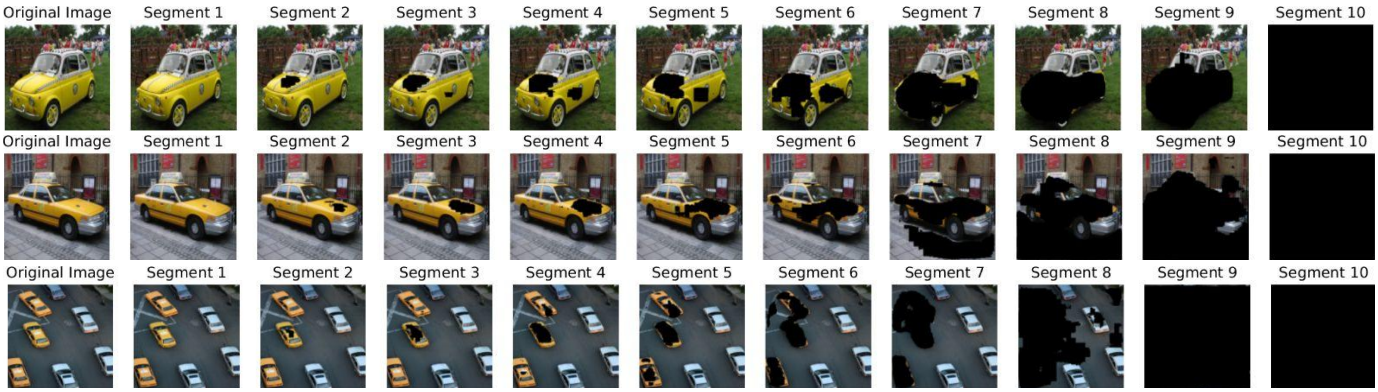

Fig. 19. The figure shows c-deletion taking place for different images from “cab” class over time for “yellow and white” seed concept.

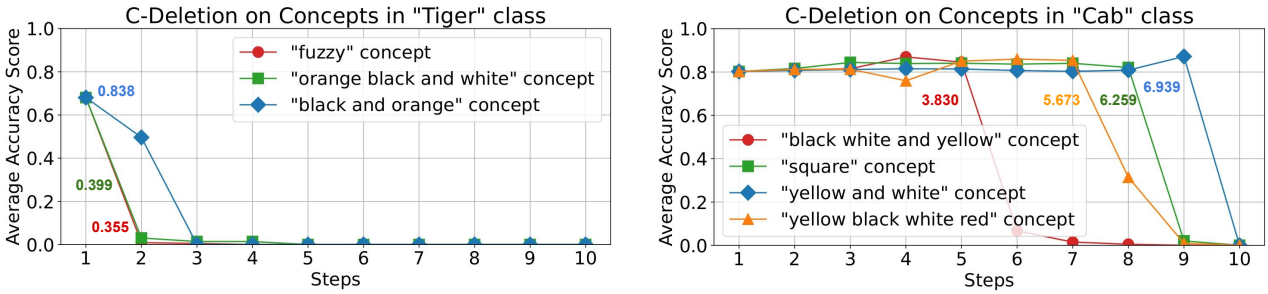

Fig. 20. C-deletion. Removing concepts over time to measure the reliability. The colored numbers indicate the area under the curve (the lower the better).

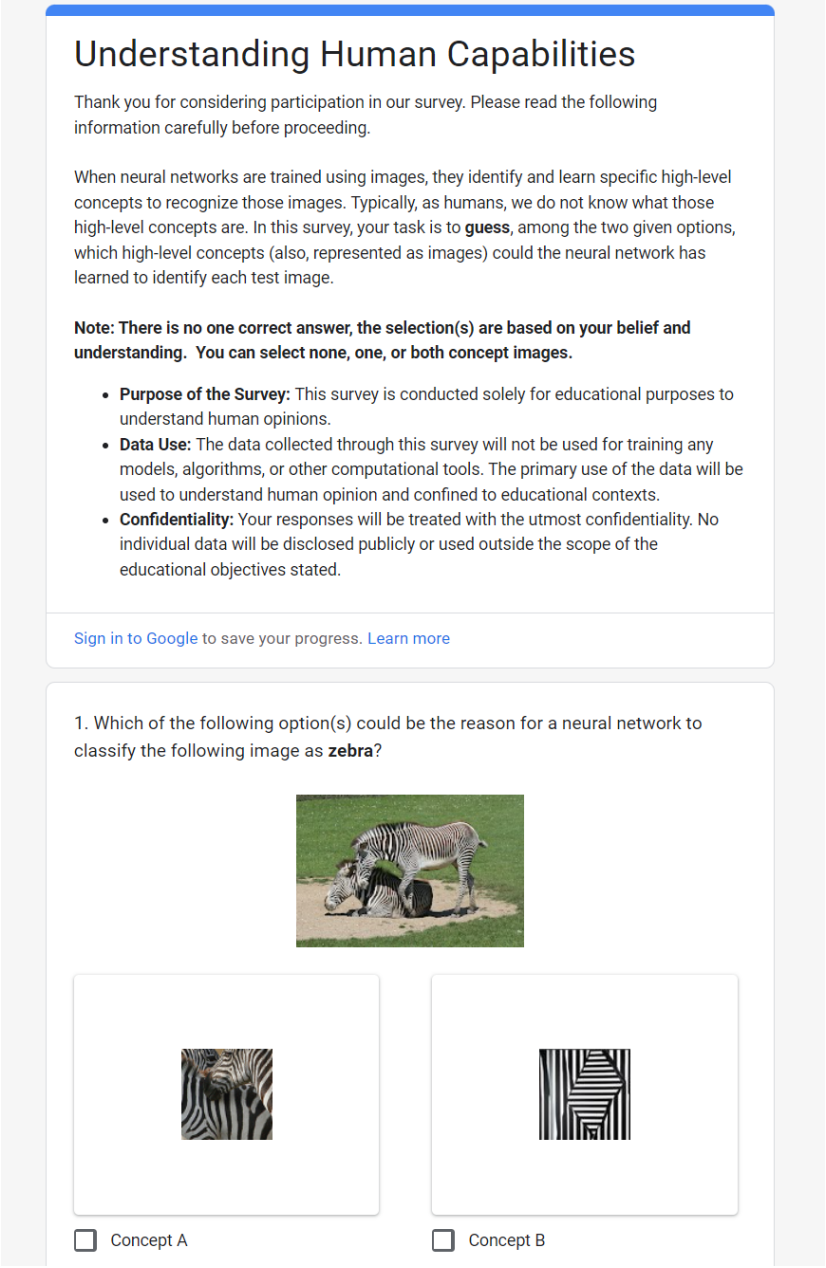

Fig. 21. A screenshot from our human survey with instructions and a sample question.

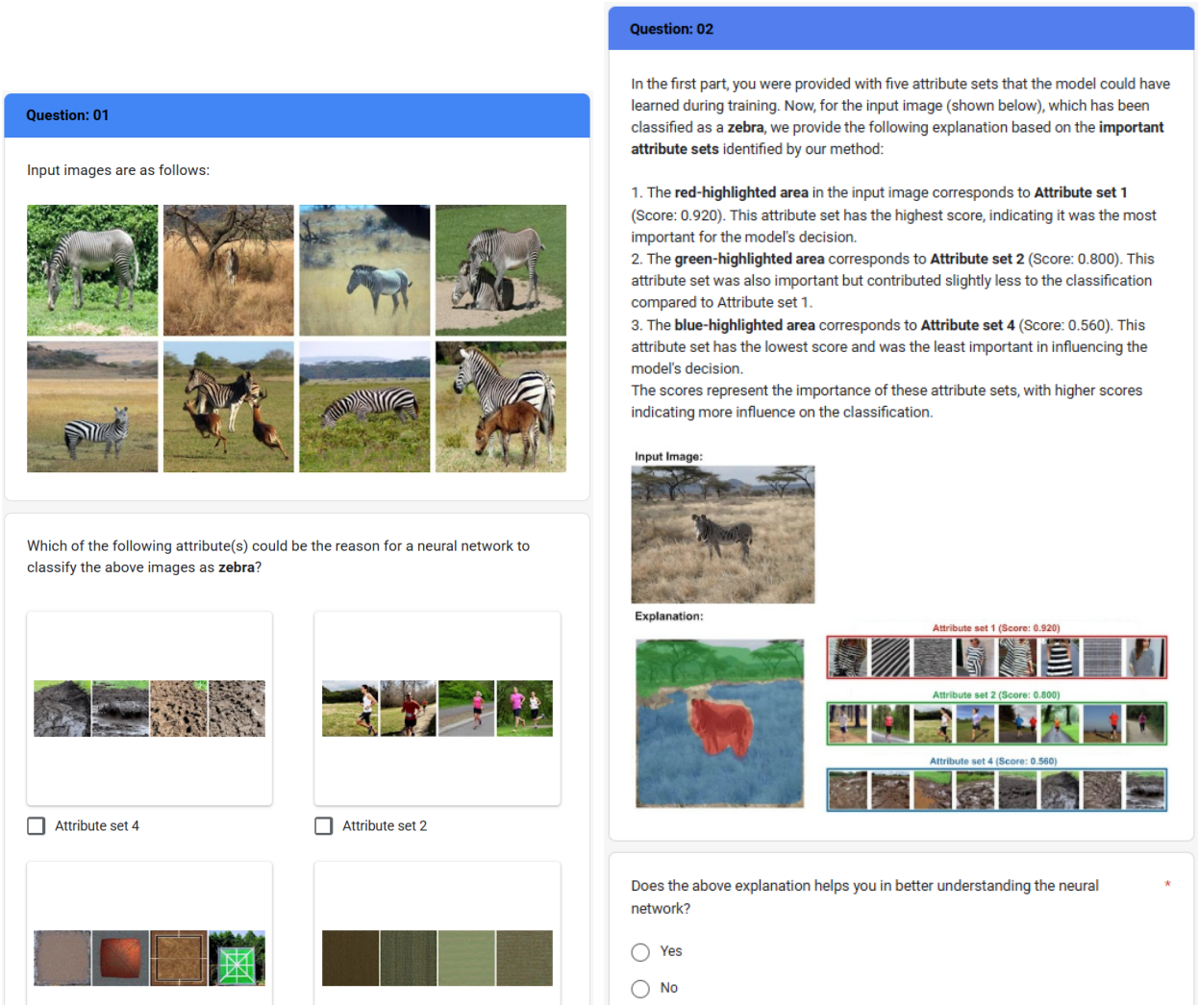

Fig. 22. Screenshots from our human survey with sample questions to validate usability.