

# Trustworthy Conceptual Explanations for Neural Networks in Robot Decision-Making

Som Sagar<sup>1\*</sup>, Aditya Taparia<sup>1\*</sup>, Harsh Mankodiya<sup>1</sup>, Pranav Bidare<sup>1</sup>, Yifan Zhou<sup>1</sup>, and Ransalu Senanayake<sup>1</sup>

**Abstract**—Black box neural networks are an indispensable part of modern robots. Nevertheless, deploying such high-stakes systems in real-world scenarios poses significant challenges when the stakeholders, such as engineers and legislative bodies, lack insights into the neural networks’ decision-making process. Presently, explainable AI is primarily tailored to natural language processing and computer vision, falling short in two critical aspects when applied in robots: grounding in decision-making tasks and the ability to assess trustworthiness of their explanations. In this paper, we introduce a *trustworthy* explainable robotics technique based on human-interpretable, high-level *concepts* that attribute to the decisions made by the neural network. Our proposed technique provides explanations with associated uncertainty scores by matching neural network’s activations with human-interpretable visualizations. To validate our approach, we conducted a series of experiments with various simulated and real-world robot decision-making models, demonstrating the effectiveness of the proposed approach as a post-hoc, human-friendly robot learning diagnostic tool. Code: <https://github.com/aditya-taparia/BaTCAVe>

## I. INTRODUCTION

A significant number of models in robotics research are now equipped with deep neural networks (DNNs), with an increasing trend towards end-to-end models [1], [2]. Nevertheless, only a few DNNs are deployed in real-world robots, and those that are deployed mostly focus on tasks such as object detection rather than decision-making or control. This hesitation to use DNNs in real robots is partly due to the high-stakes nature of robot decision-making, where it is unsafe to deploy systems without a clear understanding of their inner workings. Although we do not understand these black-box DNNs, we cannot simply discard them due to their remarkable performance in certain test cases. Hence, we advocate for developing new methods to explain how they work. To this end, instead of focusing on inherently interpretable white-box or gray-box models, this paper focuses on explaining black-box neural networks in robots post-hoc.

While there are many post-hoc explainable techniques proposed by the machine learning community [3], [4], [5], most do not focus on decision-making of a robot or a physical system. For robot decision-making, we want to explain how certain aspects of the input contribute to a particular action or a set of actions. Such explanations help engineers with debugging and legislative bodies with certifying these models. Therefore, to make explanations human-centric, we consider *concepts*—defined as high-level attributes that help humans

understand these black boxes [6]. As an example, the concept of stripes explains why a DNN would classify an image as a zebra. Concepts do not necessarily need to be pixel-level geometric patterns, as in feature attribution methods [3], [7], [4], [8], [5]. For instance, in our experiments, we will show how the concept of darkness of an object can be used to explain collision avoidance decisions of a mobile robot.

If an explainable AI technique provides an explanation about why a robot learning algorithm took a particular decision, why should human trust that explanation? Not only some explanations can be wrong but also there can be multiple explanations for the same decision. Since improving the trustworthiness of a robot explainer is crucial, we propose a method named, Bayesian Testing with Concept Activation Vectors (BaTCAVe), that assigns a score and an associated uncertainty for concepts of interest. The score indicates how well the concept explains the decision and the uncertainty indicates how much to trust the concept. To obtain these scores, we consider a posterior distribution over concept activation vectors, which, due to the non-exponential-family likelihood, is approximated using variational inference. Contributions of the paper are as follows:

- 1) Proposing a post-hoc explainable decision-making technique for robots equipped with DNNs.
- 2) Proposing a theoretical framework to evaluate the trustworthiness of explanations.
- 3) Discovering explainable concepts for diverse robotics decision-making tasks.

Given that failures are inevitable despite efforts to create robust models [9], we believe our work in developing explainability tools will assist roboticists in iteratively improving models to enhance their robustness and safety.

## II. RELATED WORK

**Explainability in robotics:** Since robotics is a high-stakes task with many modules that interact with each other, the majority of real-world robots are designed to be interpretable by construction. For instance, they are typically equipped with white-box planning [10] and control [11] algorithms. Even when models are data-driven, they tend to be constructed as gray-boxes—parameters of an inherently interpretable model are estimated using data [12], [13], [14]. As an example, by using a decision tree as the policy, a reinforcement learning algorithm can be made interpretable [12]. Unlike these methods, our focus is developing techniques to probe and explain inherently black-box neural networks in robot decision-making, whether they are used in a modular or an end-to-end fashion. Another line of research has explored

\*Equal contribution

<sup>1</sup>School of Computing and Augmented Intelligence, Arizona State University, Tempe, Arizona <ssagar6, ataparia, hmankodi, pbidare, yzhou298, ransalu>@asu.edu

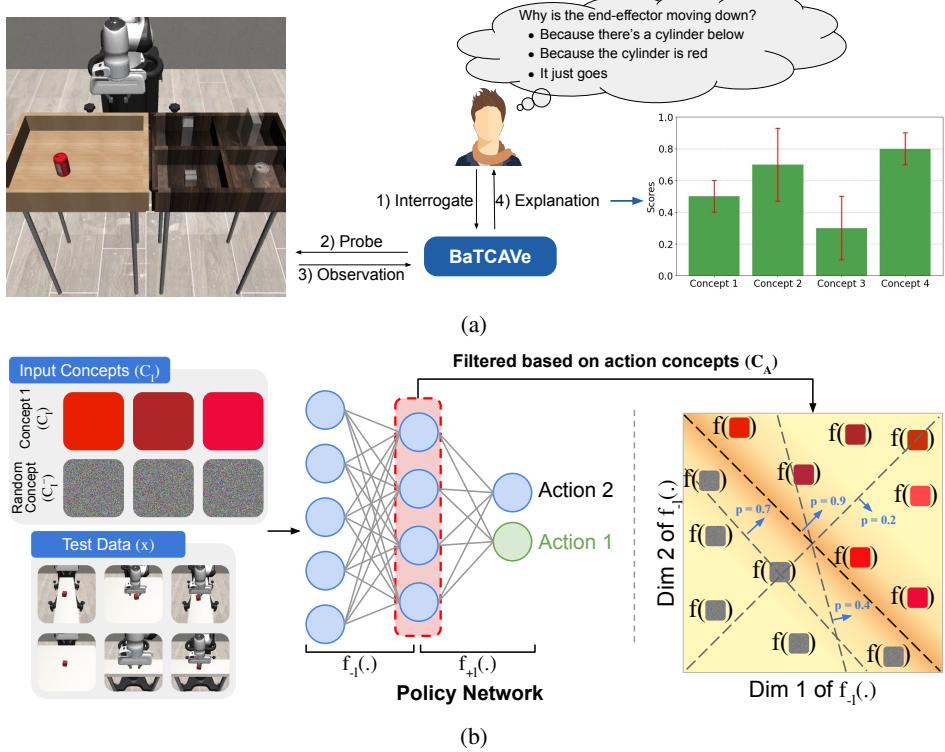


Fig. 1: Proposed explainability pipeline. a) The user obtains a score with uncertainty for each “concept.” b) The user provides input concepts  $C_I$  and test data  $x$ . The policy network  $f(\cdot)$  returns a score reflecting the alignment of the concept with the network and its uncertainty. Action concepts  $C_A$  define a subset of the output space for focused analysis, filtering actions based on user interest.

explainability of robots for their everyday human users [15], [16]. In contrast, our focus is on engineers and legislators who aim to audit robot learning models and require distinct, actionable explanations.

**Explainable AI:** Unlike inherently interpretable models, the goal of explainable AI (XAI) is to develop methods for explaining black-box models. XAI techniques focus on how to extract explanations as well as how to represent them. Some techniques achieve explainability by testing input components through perturbations [4] or component removal [7], while others use local approximations of the global decision boundary [7]. XAI methods may leverage gradients [5], weights [17], or layer-wide insights [6] to generate explanations. These explanations can be represented by highlighting specific parts of an image [5], assigning importance scores to an image segment or pixel clusters [3], [7], or by analyzing representative samples [18], [6]. In robot decision-making, highlighting certain parts of an image is not useful as it does not truly explain what aspects of the highlighted pixels is indeed important. For instance, if the XAI technique highlights a car, is it the model or the color that is important? Also, XAI methods typically do not provide uncertainty about their explanations, making them difficult to trust in high-stakes applications such as robotics. Considering the importance of trustworthiness in explanations for robots [19], our approach not only provides explanations but also quantifies their uncertainty.

**Concepts:** In machine learning, concepts are defined as human-interpretable, high-level attributes. For instance, the concept of stripes is important for a classifier to identify a zebra [6], [20]. While concept-based explanations have been applied in domains such as biomedical imaging [21], they are still applied to typical discrete image classification settings. In this paper, we propose a method to provide explanations for decisions and control commands using concepts. While a recent work discusses continuous explanations [22], it focuses on non-binary concepts rather than non-binary outputs, which are what matters in robot control. We apply the method we propose for decision networks, behavioral cloning, and deep reinforcement learning. Further, our method can provide epistemic uncertainty [23] of explanations, indicating how trustworthy an explanation is.

### III. METHOD

Similar inputs to a neural network result in similar activation patterns at a given layer. By projecting these activations into a human understandable representation, humans can explain how the neural networks work in certain aspects. To make such a projection, concept activation vectors (CAVs) can be used. They measure how sensitive a classifier’s output is to some activations in the direction of a concept of interest [6]. For instance, if the classification of a zebra image is more sensitive to some images of stripes than images of dots (i.e., the direction), then stripes play a

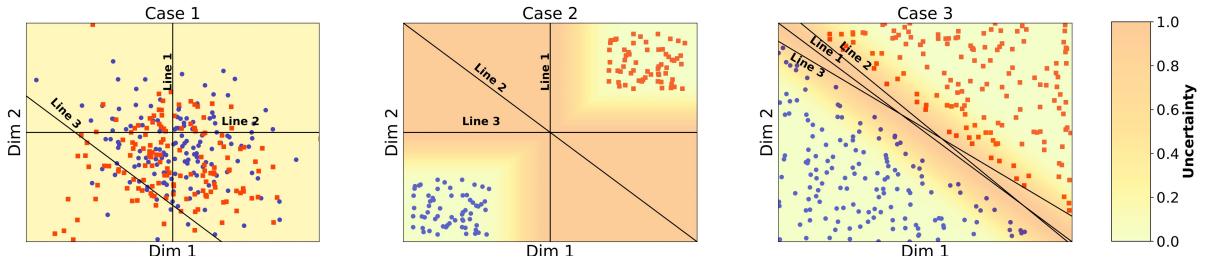


Fig. 2: Data with two classes (red and blue) are represented in the activation space. If the uncertainty is high (case 2 vs. 3), then we can sample many valid lines (i.e., many explanations). Though many lines can be sampled from case 1 as well, since the accuracy is low, the explanations cannot be trusted.

more significant role in classifying a zebra. However, this classical notion in computer vision cannot be used in robotic decision-making for several reasons. First, robot actions can contain continuous control signals, for which CAVs are not defined. Second, they do not provide uncertainty estimates of explanations. Since an explainer will always provide an explanation, we do not know whether to trust them or not without uncertainty. To resolve the first issue, we redefine the concept for robotics in the form of input concepts and action concepts, which are subsequently used to resolve the later issue using Bayesian inference, through which we can estimate the uncertainty of the explanations. As illustrated in Fig. 1, we propose a method that assigns a metric to assess how well a given concept explains robot’s decisions. This metric also comes with an estimation of epistemic uncertainty of the explanation. We propose the following definitions before presenting the method.

**Action Concepts:** Defining action concepts helps us target specific robot behavior for which we seek explanations. Consider a pre-trained neural network model,  $f(\cdot)$ , that provides outputs,  $\mathbf{y} = f(\mathbf{x})$ , with  $\mathbf{y} \in \mathbb{R}^D$ , for an input  $\mathbf{x}$ . An action concept,  $C_A$ , is a set of conditions that defines a subset of the output space a user is interested in analyzing. Formally,  $C_A = \bigoplus_{d=1}^D \mathbf{y}_d$  for  $\mathbf{y}_d \in \mathbb{R}_d \subseteq \mathbb{R}$  under logical operations,  $\oplus$ , such as conjunction ( $\wedge$ ), disjunction ( $\vee$ ), and negation ( $\neg$ ). In a manipulation example, a user might be interested in analyzing the behavior, where the end effector moves down-right while the gripper is open. In such a case, by overloading the notation  $x, y$  to represent the cardinal direction in the robot’s physical space and opened gripper distance,  $d$ , the action concepts can be defined as  $C_A = (\frac{dx}{dt} > 0) \wedge (\frac{dz}{dt} < 0) \wedge (d > 0.5)$ . Action concepts help us isolate the decisions or behaviors that we are interested in explaining.

**Input Concepts:** An input concept is a high-level, human interpretable attribute of robot inputs that the user believes is important to explain an action concept. These can be textures, colors, sizes, distances, directions, shapes, objects, etc. Input concepts can be defined based on engineers’ intuition, prior knowledge, or the test cases an engineer or a legislative body is interested in. Additionally, since input concepts can be automatically extracted [20], [24], [25], [26] or generated [26] based on inputs, we consider concept discovery as out-of-

scope for this paper. While action concepts are defined as rules, an input concept,  $C_I$ , is defined by a collection of representative inputs,  $\{\mathbf{x}_{C_I}\}_{m=1}^M$ . For instance,  $M$  images of stripes can be used to explain why an autonomous vehicle slowed down near a crosswalk. Activations at any layer can be computed by passing  $\mathbf{x}_{C_I}$  through the neural network.

#### A. Bayesian Testing with CAVs (BaTCAVe)

Given a collection of input concepts for an action concept, we now derive a score to measure how well each input concept explains the action concept. If an autonomous vehicle inadvertently failed to yield to another vehicle (i.e., action concept), we can test whether the color or the type of the other vehicle (i.e., input concepts) influenced the decision. To formally define the score, let us decompose the neural network into two segments,  $f_{-l}(\cdot)$  and  $f_{+l}(\cdot)$ , with one following the other, at the  $l^{\text{text}}$  layer as,  $\mathbf{y} = f(\mathbf{x}) = f_{+l}(f_{-l}(\mathbf{x}))$  for input  $\mathbf{x}$ . In other words,  $f_{-l}(\cdot)$  are the activations of a DNN at the  $l^{\text{th}}$  layer, which are then fed to the rest of the network,  $f_{+l}(\cdot)$ , to obtain the output,  $\mathbf{y}$ .

Since our objective is to build a relationship between the activations and human understandable concept inputs for some behaviors (i.e., action concepts), as shown in the plot of Fig. 1, we work in the space of activations (i.e., the space of  $f_{-l}(\cdot)$ ). The dimensionality of the activation space is equal to the number of nodes in the  $l^{\text{th}}$  layer. We can obtain the sensitivity of neural network decisions, conditioned by  $C_A$ , w.r.t. layer activations as  $\frac{\partial f^{C_A}(\mathbf{x})}{\partial f_{-l}(\mathbf{x})}$ . If we want to measure this sensitivity along a particular direction in the activation space, we have to compute the directional derivative. If this direction is a random variable,  $V$ , with its realizations,  $\mathbf{v}$ , is given by the stochastic directional derivative,

$$\begin{aligned} & \int \left( \lim_{h \rightarrow 0} \frac{f_{+l}^{C_A}(f_{-l}(\mathbf{x} + h\mathbf{v})) - f_{+l}^{C_A}(f_{-l}(\mathbf{x}))}{h} \cdot p(\mathbf{v}) \right) d\mathbf{v} \\ &= \int \frac{\partial f^{C_A}(\mathbf{x})}{\partial f_{-l}(\mathbf{x})} \cdot \mathbf{v} \cdot p(\mathbf{v}) d\mathbf{v}. \end{aligned} \quad (1)$$

If we build a relationships between  $V$  and  $C_I$ , we measure the sensitivity of the neural network decisions to human understandable input concepts. Since  $V$  is a random variable, it tells us about the uncertainty of this relationship. To build this relationship, let us estimate  $\mathbf{v}$  using an external

dataset containing inputs from  $C_I$ . To this end, similar to [6], we collect a concept dataset:  $X^{C_I} = \{(\mathbf{x}_m^{C_I}, z^+)\}_{m=1}^M$  with input concepts and a different set of inputs  $X^{C_I^-} = \{(\mathbf{x}_m^{C_I^-}, z^-)\}_{m=1}^M$  with the positive and negative classes labelled as  $z^+$  and  $z^-$ , respectively. The negative input concept,  $C_I^-$ , can be just another concept or a random collection of inputs, depending on what the user is interested in comparing. How these sets are selected will be described in the Experiments section. See Fig. . 1 for an illustration.

By computing  $f_{-l}(\mathbf{x}_m^{C_I})$  and  $f_{-l}(\mathbf{x}_m^{C_I^-})$ , we obtain the activation values of the concept dataset. If we can obtain a linear separation between the two classes, then the concept is more unique in the activation space. However, as we will discuss in Section III-B, separation in a high dimensional activation is subjected to uncertainty, we consider a probabilistic linear separation (i.e., we can draw many separation lines with different probabilities). By setting this probabilistic linear separation the same as  $\mathbf{v}$ , we obtain the relationship between  $V$  and  $C_I$ . To estimate the vector distribution,  $\mathbf{v}$ , that separates the two classes in the activation space, we apply the Bayes theorem,

$$\underbrace{p(\mathbf{v}|z, X^{C_I}, X^{C_I^-})}_{\text{posterior}} \propto \underbrace{p(z|\mathbf{v}, X^{C_I}, X^{C_I^-})}_{\text{likelihood}} \times \underbrace{p(\mathbf{v})}_{\text{prior}}. \quad (2)$$

Given the positive and negative labels, the likelihood follows a Bernoulli distribution,  $z|\mathbf{v} \sim \text{Bernoulli}$ . The prior weight distribution is considered to follow a normal distribution,  $\mathbf{v} \sim \mathcal{N}$ . However, because of the non-conjugate prior, the posterior distribution is not tractable [23], [27]. Hence, we resort to approximate Bayesian inference. Because the activation space of the neural network can be high dimensional, rather than using an MCMC technique, we use variational inference, where we minimize the KL divergence between the true posterior and an approximate posterior distribution,  $q(\mathbf{v})$ . However, since the true posterior is not known, instead of minimizing the KL divergence, we maximize an evidence lower bound (ELBO),

$$\text{ELBO} + \mathbb{KL}[q(\mathbf{v})||p(\mathbf{v}|z, X^{C_I}, X^{C_I^-})] = \text{const.} \quad (3)$$

Following the locally linear approximation of the posterior in [28], we learn  $q(\mathbf{v})$  in an expectation-maximization-style.

On a collection of test inputs,  $X^{\text{test}} = \{\mathbf{x}_p^{\text{test}}\}_{p=1}^P$ , with relation to (1), we can now compute a score,  $s^{C_A, C_I, C_I^-}$ ,

$$\frac{1}{|X^{\text{test}}|} \sum_{\mathbf{x}^{\text{test}} \in X^{\text{test}}} \mathbb{I} \left( \left( \int \frac{\partial f^{C_A}(\mathbf{x}^{\text{test}})}{\partial f_{-l}(\mathbf{x}^{\text{test}})} \cdot \mathbf{v} \cdot q(\mathbf{v}) d\mathbf{v} \right) > 0 \right) \quad (4)$$

This score itself is a distribution. If we obtain samples from  $q(\mathbf{v})$ , each of them is a valid explanation. Since some samples are more probable than others, those with high probability are more likely concepts that explain the decisions. The empirical mean and standard deviation of the score can be estimated easily. The mean score  $\bar{s}$  is given by,

$$\frac{1}{R \cdot |X^{\text{test}}|} \sum_{\mathbf{x}^{\text{test}} \in X^{\text{test}}} \sum_{\mathbf{v}_r \in V^{C_I}} \mathbb{I} \left( \frac{\partial f^{C_A}(\mathbf{x}^{\text{test}})}{\partial f_{-l}^{C_A}(\mathbf{x}^{\text{test}})} \cdot \mathbf{v}_r > 0 \right) \quad (5)$$

where  $V^{C_I} = \{\mathbf{v}_r\}_{r=1}^R$  are  $R$  samples taken from  $q(\mathbf{v})$ . The higher the score, the better the concept  $C_I$  explains the test inputs  $X^{\text{test}}$ . Similarly, the empirical standard deviation reflects the epistemic nature [23] of explanations—how much the model knows that its explanation can be wrong.

### B. Interpreting the Trustworthiness of Explanations

To assess the trustworthiness of explanations, relying on 1) the held-out test accuracy from eq. (2) and 2) the epistemic uncertainty of the score in eq. (4), we consider (Fig. 2):

**Case 1** (Off-base explanations): If the accuracy is low, the concepts are not good enough to delineate the CAVs, resulting in an inaccurate explanation. Formally, in such cases,  $p^{C_I} \approx p^{C_I^-}$ , where the distributions are defined as  $X^{C_I} \sim p^{C_I}$  and  $X^{C_I^-} \sim p^{C_I^-}$  for the positive and negative classes, respectively. Such explanations should not be trusted.

**Case 2** (Imprecise explanations): If the accuracy is high but the uncertainty is also high, multiple explanations are possible. This can be due to, 1) the two supposedly opposite concepts are not sufficiently different enough to be delineated with a low uncertainty (i.e., some conflicting information) or, most likely, 2) the test samples lack diversity in the activation space. Unfortunately, Bayesian inference cannot differentiate lack of information from conflicting information [29].

**Case 3** (Precise explanations): If the accuracy is high and the uncertainty is low, then the concepts we have chosen are good and the explainer is able to provide consistently good explanations. These explanations are highly trustworthy.

The thresholds for probability should be decided by the practitioners depending on how much risk they are willing to take. For instance, if an engineer is using BaTCAVe to debug a manipulator used in an assembly line, the threshold can be selected leniently as the stakes might be relatively low. In such cases, we can obtain more valid explanations. In contrast, if a legislative body is using BaTCAVe to approve a new autonomous vehicle, then the thresholds should be strict. If case 1 violates, we should try new concepts to obtain a better accuracy. If case 2 violates, we can still use explanations but they might not always be the best explanations. By obtaining the mean score, we can obtain an average explanation.

## IV. EXPERIMENTS



Fig. 3: a) JetBot b) Lift cube c) Pick-and-place d) Nut assembly

We consider three types of algorithms for learning robot decision-making: binary supervised learning, behavioral cloning (BC), and proximal policy optimization (PPO). The experimental setup (also available on Github), results, and findings are detailed below.

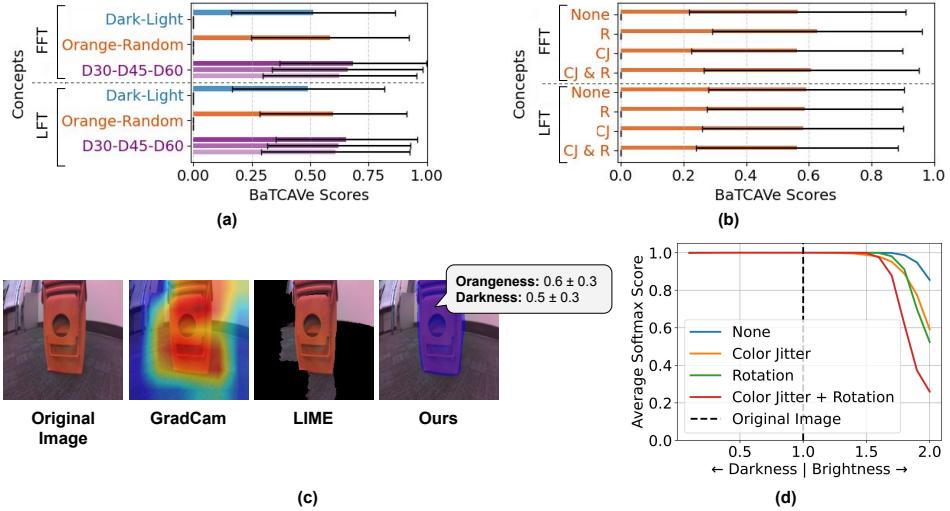


Fig. 4: The effects of (a) fine-tuning and (b) data augmentation. The higher the score, the better the explanation is. (c) Shows while common XAI methods can highlight the orange box, they do not reveal what attributes of the box contribute to the decision of the DNN, making it harder for the engineers to improve the DNN based on the explanations. In contrast, BaTCAVe provides semantically meaningful explanations. (d) Highlights the change in confidence over modifying darkness factor in input with models trained with different data augmentation (C-Modification).

#### A. Experiment 1: Mobile Robot Navigation Using Vision

**Setup:** We used a JetBot with a NVIDIA Jetson Nano (Fig 3a). It uses a pretrained AlexNet [30] with the last layer replaced with two nodes. Using 175 “obstacle” and 175 “free” images, we fine-tune the DNN so that the robot can learn to avoid obstacles. We analyze how the decision of the network is affected by different data pre-processing and fine-tuning techniques for  $C_A = \{\text{avoid obstacle}\}$  and  $C_I = \{\text{orangeness, darkness, distance}\}$ .

**Results:** With the objective of avoiding orange obstacles, we fine-tuned the full DNN with images of an orange box, shown in Fig. 3a, until we achieve a validation accuracy of 100%. We surveyed 20 engineering students who have at least 1 year of experience training DNNs to get their opinion on what the DNN has learned just by reading our description. We described them the architecture and showed fine-tuning images without telling them that our objective is avoiding orange obstacles. First, we asked them to describe what attributes the DNN should have learned and then we gave them a list of potential concepts to narrow down their choices. Many human speculated that the model will be fine-tuned to distinguish orange from the rest.

In contract, showcasing the BaTCAVe’s ability to provide true explanations that engineers might not even think of, as shown in Fig. 4(a), BaTCAVe revealed that the model has learned to distinguish dark from light objects (or the *value* or brightness in HSV scale). The orange box is merely a shade of the broader “dark” concept. Similar to C-deletion in XAI literature [20], [24], [25], we verified that the darkness is an important concept by gradually varying the brightness of the orange object, as shown in Fig. 4(d). Additionally, based on Fig. 4, BaTCAVe made the following explanations:

- The score is proportional to the concept of “distance

between the robot and obstacle,” verifying that the DNN has learned the distance.

- When the final layer is fine-tuned (LFT) instead of the full DNN (FFT), the prominence of the orange concept becomes higher compared to the dark concept, which is what we originally intended, demonstrating how engineers can use concepts to debug robots.
- As shown in Fig 4(b), the importance of the dark concept remains consistent across different data augmentation methods—adding color jitter (CJ) and/or image rotation (R)—for LFT while it varies for FFT.

Since XAI methods are hard to quantitatively benchmark, similar to other work, Fig. 4(c) qualitatively compares why BaTCAVe is a better choice than feature attribution methods for robot learning.

#### B. Experiment 2: Lift Cube, Pick & Place, and Nut Assembly with Proprioceptive Sensors

**Setup:** We use a Panda, a 6-DOF robotic arm with a gripper, to complete the three tasks shown in Fig 3(b,c,d) on RoboSuite [31]. The setup includes various proprioceptive sensors to monitor the arm’s movements and positions. By using the DNN shown in Fig 5 and the train using the ph low dim dataset from robomimic [32], we developed an agent based on behavioral cloning [32]. Robot’s actions are  $\Delta$  differences. Our objective is to explain which concepts of the inputs,  $C_I = \{\text{object, EEF position measurement, EEF quaternion measurement, gripper open width}\}$ , is responsible when the robot is taking a set of actions defined by  $C_A = \text{top } 75\% \text{ of each } \Delta \text{ action}$ .

**Results:** We obtain notably high BaTCAVe score with a low uncertainty, corroborating that accurate object information from proprioceptive sensors, unlike in vision-based

TABLE I: BaTCAVe scores across different tasks along with uncertainty estimates. Each score measures the impact of  $C_I$ .

Task	Action Concept	object	eef_pos	eef_quat	gripper
Lift Cube	$\Delta X (\uparrow)$	$0.84 \pm 0.31$	$0.42 \pm 0.40$	$0.42 \pm 0.49$	$0.42 \pm 0.49$
	$\Delta Z (\uparrow)$	$0.98 \pm 0.10$	$0.99 \pm 0.04$	$0.42 \pm 0.49$	$0.39 \pm 0.48$
Pick & place	$\Delta X (\uparrow)$	$1 \pm 0.0$	$0.54 \pm 0.49$	$0.96 \pm 0.16$	$0.09 \pm 0.09$
	$\Delta Z (\uparrow)$	$0.9 \pm 0.17$	$0.85 \pm 0.35$	$0.77 \pm 0.41$	$0.91 \pm 0.27$
Nut Assembly	$\Delta X (\uparrow)$	$0.64 \pm 0.47$	$0.59 \pm 0.49$	$0.44 \pm 0.49$	$0.87 \pm 0.32$
	$\Delta Y (\uparrow)$	$0.87 \pm 0.32$	$0.59 \pm 0.48$	$0.46 \pm 0.49$	$0.37 \pm 0.48$

settings, helps with precisely performing the task. Table I shows sample scores of  $C_A$ 's tested across different  $C_I$ 's. Further, an analysis of per time step explanations for the lifting cube task, depicted in Fig 5b, explains that only the object pose and EEF position measurements matters when the EEF is moving down.

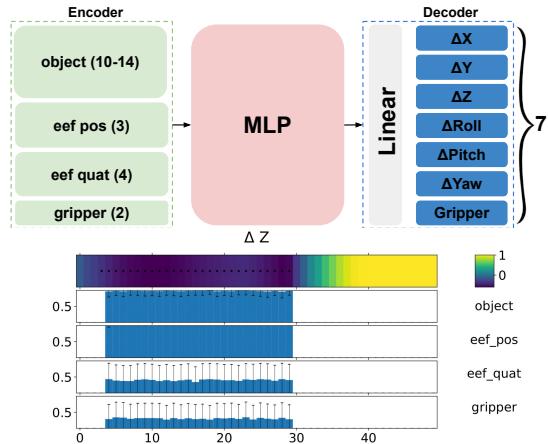


Fig. 5: a) Architecture. b) Task : Lift,  $C_A = \Delta Z$ . The colorbar indicates the  $\Delta$  actions and the dots on them indicate the actions we consider by following the rule  $C_A = \text{top 25\%}$  of moving down. The three bar plots indicate the BaTCAVe scores with their uncertainties for three input concepts EEF position, quaternion, and gripper status.

### C. Experiment 3: Lift Cube with Vision-Language Inputs

**Setup:** This experiment replicates the setup and conditions of Experiment IV-B, but proprioceptive sensors are replaced by a camera and the object information is not given to the model. The architecture is shown in Fig 6.

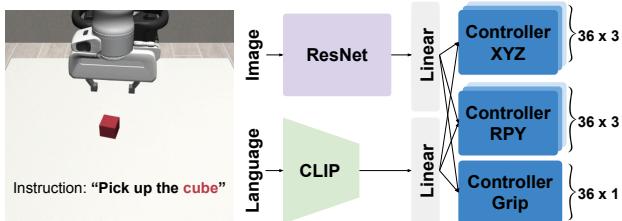


Fig. 6: Model Architecture

**Results:** We evaluated  $C_I = \{\text{images}=\{\text{cube, gripper, table}\}, \text{language}=\{\text{proper language commands, gibberish, verbs}\}\}$ . Interestingly, language concepts reveals that the verb in the sentence matters more than the rest of the sentence at the time it lifts the object.

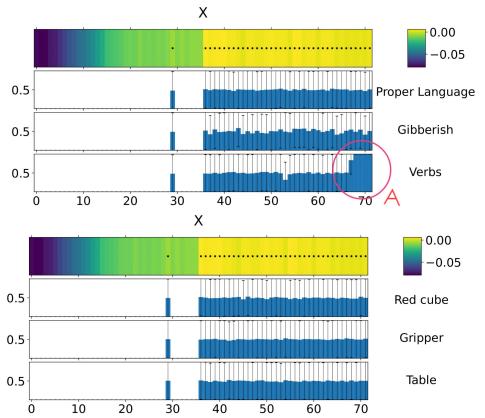


Fig. 7: (a) Indicates verbs in a language command matter most when it is lifting. (b) Indicates the images concepts are highly uncertain.

### D. Experiment 4: Autonomous Driving with Vision

**Setup:** To simulate autonomous driving, we trained an end-to-end deep reinforcement learning policy using proximal policy optimization (PPO) [33] to steer and throttle a vehicle on Donkey Simulator. The network is a CNN followed by an MLP, trained end-to-end.

**Results:** We analyzed how various input concepts affect steering decisions ( $C_A = \text{steering angle} > 2$ ). Fig. 8 shows how the input concept of “black shades,” which correlates with roads, explains steering decisions. Around  $t = 35$ , the car points out of the road, and when it steers back we observe that its BaTCAVe for “black shades” increases around  $t = 41$ , indicating that it was able to recover by focusing on the road. To improve the policy, it is important to identify what part of the neural network learns incorrect information. By applying BaTCAVe on the last CNN layer ( $\bar{s} = 0$ ) and last MLP layer ( $\bar{s} = 0.822$ ), we found that errors do not originate in the CNN, which implicitly acts as an image encoder. Therefore, instead of improving the encoder, the policy needs to be trained more.

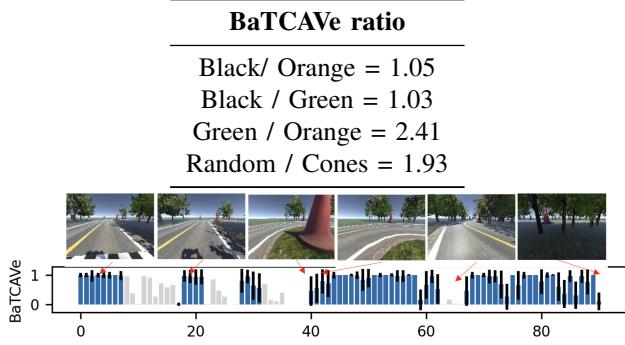


Fig. 8: (a) The relative importance of concepts indicates that the agent relies on the black concept (~ road) than the orange concept (~ cones). (b) Temporal change of scores. Scores for  $C_A$  in blue.

## V. LIMITATIONS

We cannot provide formal guarantees that BaTCAVe will surely explain because the quality of explanations depends on the concepts we decide to test. We plan to develop automatic concept discovery methods [20], [26] tailored to robotics. Further, certain concepts are difficult to test. For instance, if we test the concept of “farther away cones” in experiment 4, the activation strength is not strong enough as only a few pixels contain the signal.

## VI. CONCLUSION

We proposed a task-agnostic explainable robotics technique. We demonstrated how our method can be used to explain various robot behaviors across a variety of domains with different decision networks. The actionable insights provided by BaTCAVe helps engineers identify vulnerabilities of various components of robot training—data augmentation, fine-tuning, domain-shift analysis, verification, etc. We showed how uncertainty that BatCAVe provides helps with trusting explanations. Future work will focus on developing concept dictionaries for different robotic tasks.

## REFERENCES

- [1] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan *et al.*, “Open x-embodiment: Robotic learning datasets and rt-x models,” *arXiv preprint arXiv:2310.08864*, 2023.
- [2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, “Open-vla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.
- [3] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘why should i trust you?’ explaining the predictions of any classifier,” in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
- [4] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.
- [5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
- [6] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas *et al.*, “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav),” in *International conference on machine learning*. PMLR, 2018, pp. 2668–2677.
- [7] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.
- [8] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: High-precision model-agnostic explanations,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
- [9] S. Sagar, A. Taparia, and R. 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.
- [10] M. Fox, D. Long, and D. Magazzeni, “Explainable planning,” *arXiv preprint arXiv:1709.10256*, 2017.
- [11] D. Bertsekas, *Reinforcement learning and optimal control*. Athena Scientific, 2019, vol. 1.
- [12] S. Milani, Z. Zhang, N. Topin, Z. R. Shi, C. Kamhoua, E. E. Papalexakis, and F. Fang, “Interpretable multi-agent reinforcement learning with decision-tree policies,” in *Explainable Agency in Artificial Intelligence*. CRC Press, 2024, pp. 86–120.
- [13] S. Milani, N. Topin, M. Veloso, and F. Fang, “A survey of explainable reinforcement learning,” *arXiv preprint arXiv:2202.08434*, 2022.
- [14] R. P. Bhattacharyya, R. Senanayake, K. Brown, and M. J. Kochenderfer, “Online parameter estimation for human driver behavior prediction,” in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 301–306.
- [15] A. Silva, P. Tambwekar, M. Schrum, and M. Gombolay, “Towards balancing preference and performance through adaptive personalized explainability,” in *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 658–668.
- [16] D. Das, S. Banerjee, and S. Chernova, “Explainable ai for robot failures: Generating explanations that improve user assistance in fault recovery,” in *Proceedings of the 2021 ACM/IEEE international conference on human-robot interaction*, 2021, pp. 351–360.
- [17] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, “Network dissection: Quantifying interpretability of deep visual representations,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6541–6549.
- [18] P. W. Koh and P. Liang, “Understanding black-box predictions via influence functions,” in *International conference on machine learning*. PMLR, 2017, pp. 1885–1894.
- [19] L. Sanneman and J. A. Shah, “Trust considerations for explainable robots: A human factors perspective,” *arXiv preprint arXiv:2005.05940*, 2020.
- [20] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, “Towards automatic concept-based explanations,” *Advances in neural information processing systems*, vol. 32, 2019.
- [21] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, “Concept bottleneck models,” in *International conference on machine learning*. PMLR, 2020, pp. 5338–5348.
- [22] M. Graziani, V. Andreadczyk, and H. Müller, “Regression concept vectors for bidirectional explanations in histopathology,” in *Understanding and Interpreting Machine Learning in Medical Image Computing Applications: First International Workshops, MLCN 2018, DLF 2018, and iMIMIC 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings 1*. Springer, 2018, pp. 124–132.
- [23] R. Senanayake, “The role of predictive uncertainty and diversity in embodied ai and robot learning,” in *arXiv preprint arXiv:2405.03164*, 2024.
- [24] T. Fel, A. Picard, L. Bethune, T. Boissin, D. Vigouroux, J. Colin, R. Cadène, and T. Serre, “Craft: Concept recursive activation factorization for explainability,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2711–2721.
- [25] T. Fel, V. Boutin, L. Béthune, R. Cadène, M. Moayeri, L. Andéol, M. Chalvidal, and T. Serre, “A holistic approach to unifying automatic concept extraction and concept importance estimation,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [26] A. Taparia, S. Sagar, and R. Senanayake, “Explainable concept generation through vision-language preference learning,” 2024.
- [27] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.
- [28] T. S. Jaakkola and M. I. Jordan, “A variational approach to Bayesian logistic regression models and their extensions,” in *Proceedings of the*

- Sixth International Workshop on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, D. Madigan and P. Smyth, Eds., vol. R1. PMLR, 04–07 Jan 1997, pp. 283–294.
- [29] G. J. Klir and A. Ramer, “Uncertainty in the dempster-shafer theory: a critical re-examination,” *International Journal of General System*, vol. 18, no. 2, pp. 155–166, 1990.
  - [30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, 2012.
  - [31] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu, “robosuite: A modular simulation framework and benchmark for robot learning,” in *arXiv preprint arXiv:2009.12293*, 2020.
  - [32] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, “What matters in learning from offline human demonstrations for robot manipulation,” in *arXiv preprint arXiv:2108.03298*, 2021.
  - [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

## APPENDIX

### I. EXPERIMENT 1: MOBILE ROBOT NAVIGATION

In this experiment we considered two distinct setups. In the initial setup we trained the obstacle classifier specifically on an orange box, varying in different orientation and distance, as shown in Fig 9. We then compare the variation in importance of concepts caused by the fine-tuning method used and the pre-processing steps involved during training. In the second setup, we generalized the classifier by introducing different objects as obstacles in the training dataset. We test the model with color, darkness and distance as a concept. These criteria were selected based on a human study(Appendix I-C) performed where we ask them to describe important attributes the model might have learned given the dataset. The dataset used to train the classifier model which is used for decision making in the JetBot is collected from the onboard camera attached on top the JetBot. In both the setups, we have 350 images in training and 150 in testing, split equally among both classes as shown in Fig 10.



Fig. 9: Snapshot of JetBot rollout

**Jetbot configuration:** The JetBot is a compact robot built on the NVIDIA Jetson Nano platform, designed for AI and robotics applications. Key components of the JetBot include: The Jetson Nano board which has a 4GB LPDDR4 RAM, a 128-core NVIDIA Maxwell GPU, and a quad-core ARM Cortex-A57 MPCore processor. An 8 MP wide-angle camera is attached on top. It also includes two motors and a motor driver for precise control of the robot's wheel movement.

**AlexNet** [30]: AlexNet is the classifier model we use for decision making in JetBot. AlexNet was used due its high processing speed which is critical in robotics. We change the architecture by replacing the final layer with a layer with 2 nodes.

**Training:** We finetuned the AlexNet model with orange box obstacle dataset and general object obstacle dataset. We finetuned the models with FFT and LFT for 100 epochs with cross-entropy loss and SGD optimizer with learning rate of 0.001 and momentum 0.9, while applying color jitter (CJ) and/or image rotation (R). The training loss across different experiments are shown in Fig 12.



Fig. 10: Orange box Obstacle and Not obstacle dataset sample

### A. Concepts

Fig 11 shows the concepts which were used across different experiments. Random concepts were creating by randomly picking a unique color for each pixel value in the image. We evaluated the models with dark-light concept, orange-random concept, distance 30-random concept, distance 45-random concept, and distance 60-random concept sets.

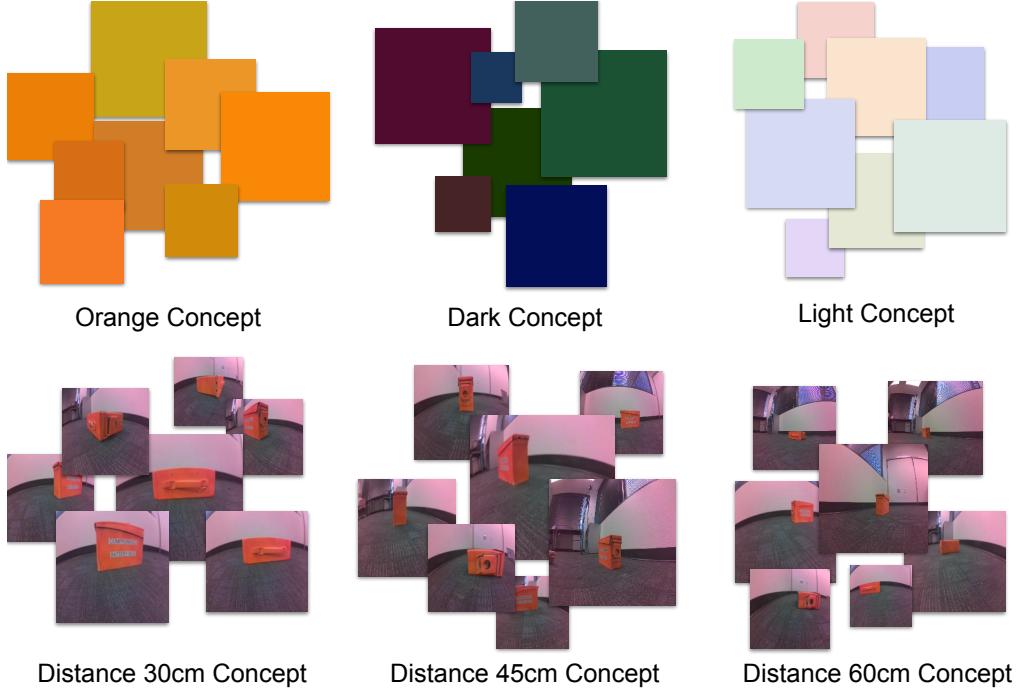


Fig. 11: Different concepts sets used in BaTCAVe for Experiment 1.

### B. Additional Results

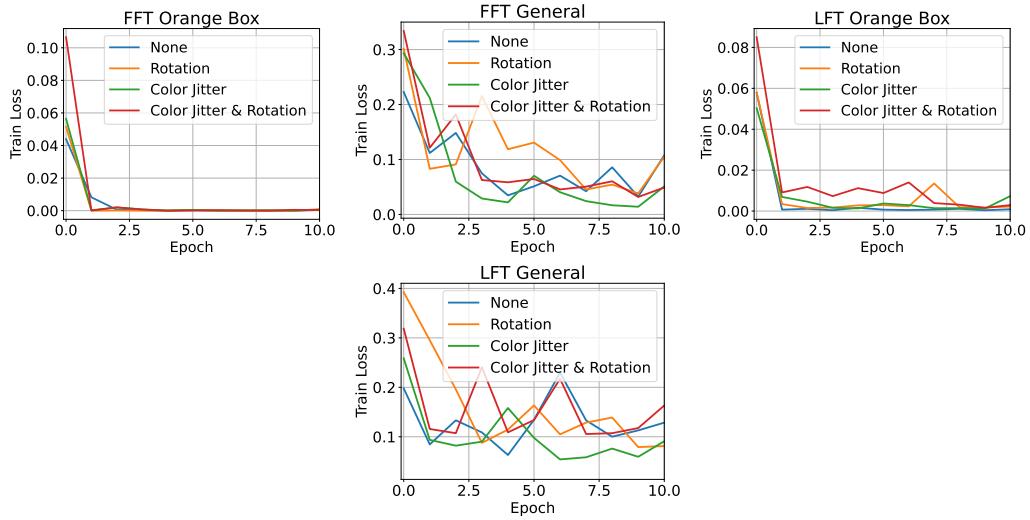


Fig. 12: Training loss of FFT (left) and LFT (right) over orange obstacle and general obstacle dataset and different augmentations.

Fig 13 and Fig 14 shows the results of general model, finetuned with all parameters and final layer parameters, respectively. They are compared over different concepts and augmentation techniques. We see that in both the cases models were not able to understand dark difference between dark and light concept.

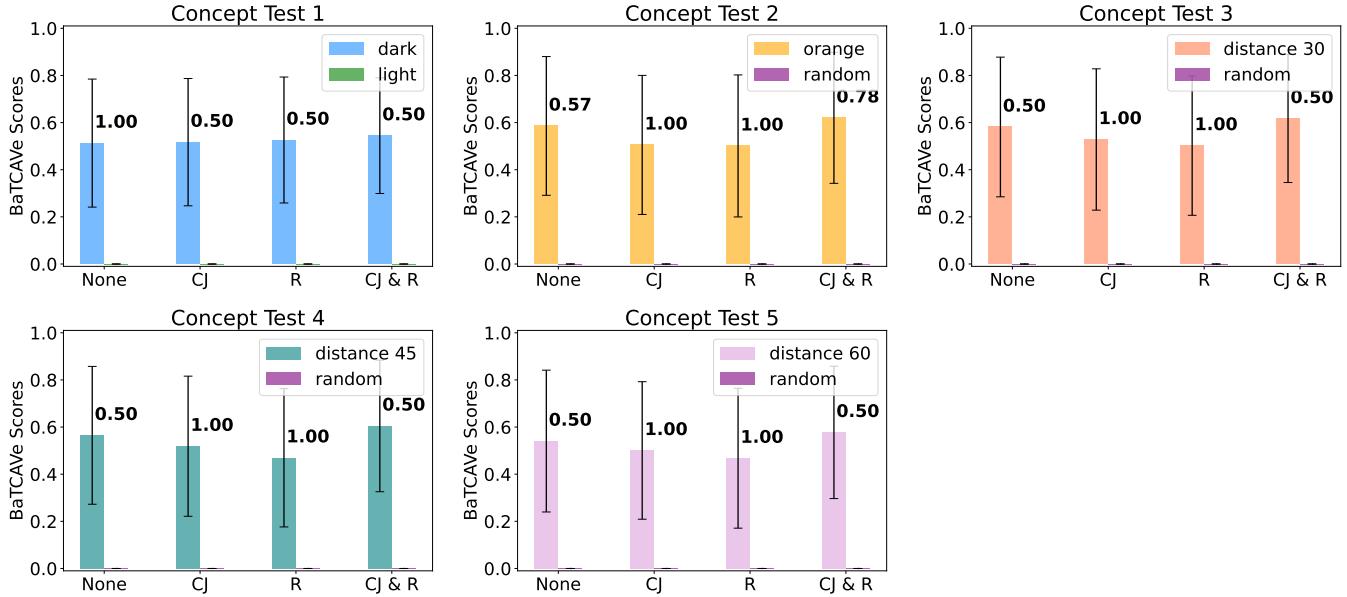


Fig. 13: Different concepts test on FFT model trained on general dataset over different augmentations. (BaTCAVe’s classifier accuracy is shown near the bar graph for each augmentation.)

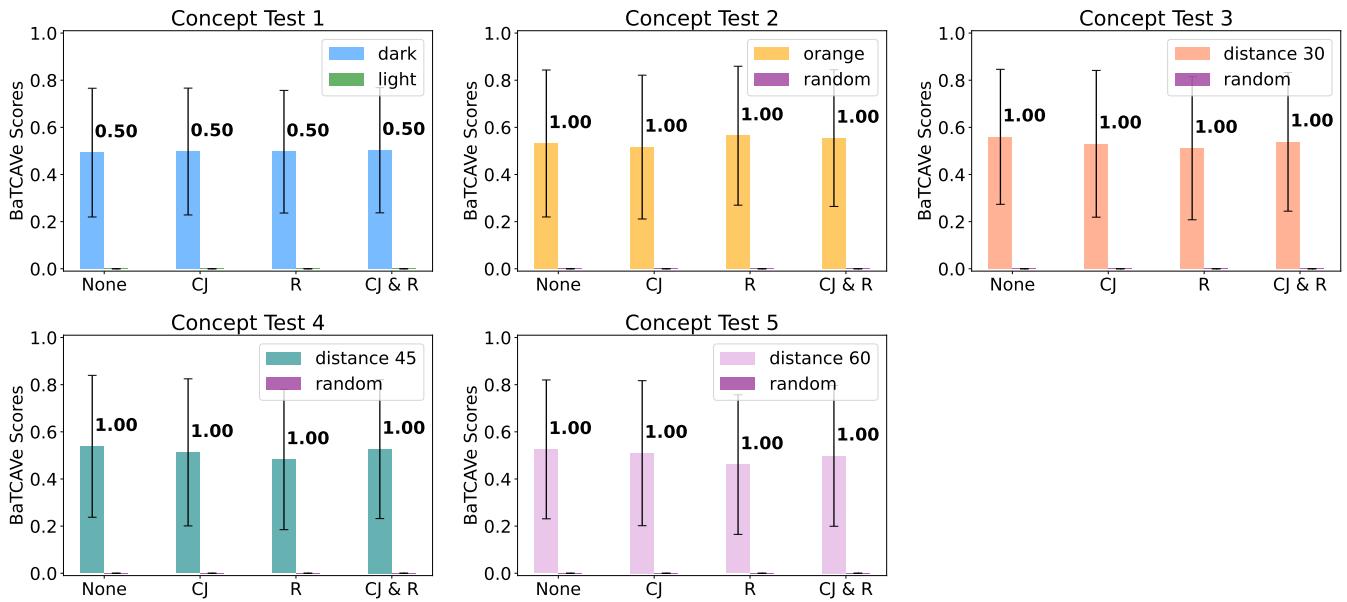


Fig. 14: Different concepts test on LFT model trained on general dataset over different augmentations. (BaTCAVe’s classifier accuracy is shown near the bar graph for each augmentation.)

### C. Human survey

Fig 15(a) shows the screenshot of the survey that was circulated. Fig 15(b) is the distribution of  $C_I$  chosen by the participants (20). Fig 15(c) shows the word cloud representation of all the replies gotten by the participants.

## II. EXPERIMENT 2: TASKS WITH PROPRIOCEPTIVE SENSORS

### A. Model details

**Architecture:** The model architecture, depicted in Fig 5(a), employs a low-dimensional observation modality that integrates multiple sensor inputs: gripper position (2-vector), end-effector position (3-vector), object characteristics (10-14 vector), and end-effector orientation (4-vector). The network outputs a 7-dimensional action vector which includes delta values for the robot’s movements: three for end-effector position (x, y, z), three for orientation (roll, pitch, yaw), and one for gripper force.

The model has 3 main components: Observation Encoding, Multi-Layer Perceptron(MLP), Observation Decoding.

What NN learns and what humans think

Sign in to Google to save your progress. [Learn more](#)

\* Indicates required question

**Question: 01**

We are using a modified AlexNet to make decisions in the Jetbot robot for navigation and obstacle avoidance. This modification is made by replacing the last layer of a pre-trained AlexNet (architecture shown below) with 2 output nodes to build a binary classifier to classify input images as **obstacle** or **not obstacle**.

**AlexNet**

```

Image: 224 (height) x 224 (width) x 3 (channels)
Convolution with 11x11 kernel+4 stride: 54x54x96
    | ReLU
    | Pool with 3x3 max kernel+2 stride: 26x26x96
    |
    | Convolution with 5x5 kernel+2 pad: 26x26x256
    |    | ReLU
    |    | Pool with 3x3 max kernel+2 stride: 12x12x256
    |
    | Convolution with 3x3 kernel+1 pad: 12x12x384
    |    | ReLU
    |    | Convolution with 3x3 kernel+1 pad: 12x12x384
    |        | ReLU
    |        | Convolution with 3x3 kernel+1 pad: 12x12x256
    |            | ReLU
    |            | Pool with 3x3 max kernel+2 stride: 5x5x256
    |                | flatten
    |                | Dense: 4096 fully connected neurons
    |                    | ReLU, dropout p=0.5
    |                    | Dense: 4096 fully connected neurons
    |                        | ReLU, dropout p=0.5
    |                        | Dense: 1000 fully connected neurons
    |
    | Output: 1 of 1000 classes

```

We then fine-tune the whole network with images similar to the one shown below \* (175 images for each class x 2 classes = 350 images) until the validation accuracy is close to one.

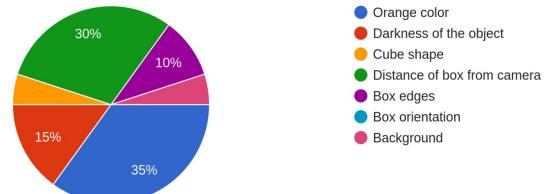
Based on the dataset sample provided, describe in short phrases (less than 250 characters), what features could the neural network has learned to classify the images as **obstacle**?

**Obstacle**

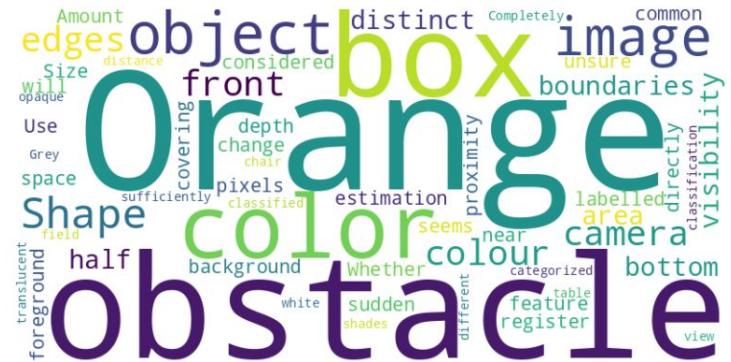


a)

For the previously described network, which of the following attributes could the classifier have learned to classify images as **obstacle**? Note: Please select the most important factor from the options below.  
20 responses



b)



c)

Fig. 15: Human Survey: a) Survey screenshot b) Percentage of concept chosen c) Wordcloud of response given

- 1) Observation Encoding: Prior to processing, observed states undergo encoding through an Observation Group Encoder. This encoder handles observation modalities, which are object information, robot end effector position and orientation, and gripper state. The encoded observations are concatenated into a single vector representation.
- 2) MLP: The concatenated vector is then fed into an MLP comprising a single hidden layer with ReLU activation, facilitating the extraction of high-level features from the encoded observations. The output dimensionality of the MLP is 1024.
- 3) Observation Decoding: Following feature extraction, the output of the MLP is decoded to produce the desired actions. This decoding process involves a linear transformation, mapping the MLP output to the action space. The actions consist of seven elements, corresponding to the changes in position ( $x$ ,  $y$ ,  $z$ ) and orientation (roll, pitch, yaw) of the robot end effector, as well as the gripper state.

**Training:** We train the model using the ph\_low\_dim dataset from robomimic [32] on task can,lift and square. The training process begins with data normalization to ensure consistent input scaling. The training loop spans 500 epochs, each consisting of 100 gradient steps. During each step, the model calculates the mean squared error (MSE) loss between predicted and ground truth actions. This loss is then used to update the model parameters using the Adam optimizer.

This model was trained across three different tasks:

- 1) Pick Place Can(PPC): The primary goal of this task is for the robot to accurately pick up a can from one location using a gripping mechanism and then move it to a specified area to place it down. Fig 17 shows the state across different timesteps and Fig 18 shows BaTCAVe performed across multiple  $C_A$  on PPC.
- 2) Nut Assembly(NA): This task requires the robot to pick up a hollow object and accurately pick its handle and fit it into the object which fits the hollow area. Fig 19 shows the state across different timesteps and Fig 20 shows BaTCAVe

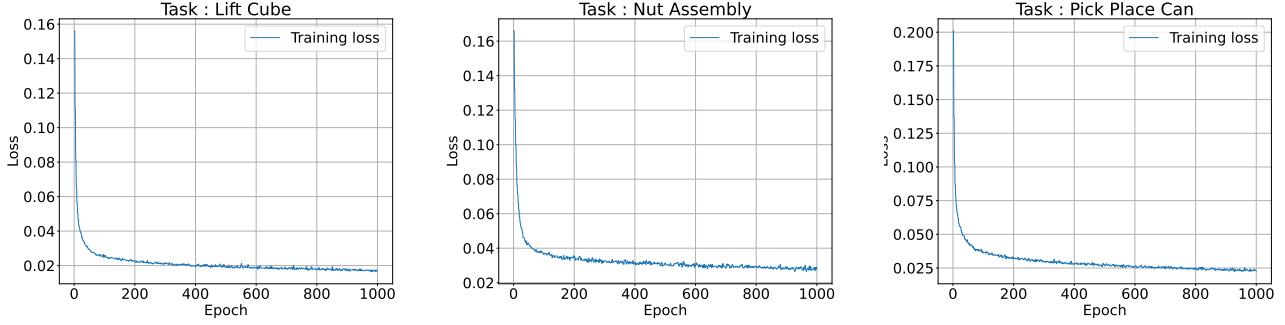


Fig. 16: Training loss of model across tasks

performed across multiple  $C_A$  on NA.

- 3) Lift Cube(LC): In the LC task, the robot's goal is to lift a cube from the table. Fig 21 shows the state across different timesteps and Fig 22 shows BaTCAVe performed across multiple  $C_A$  on LC.

Table II shows BeTCAVe scores across all tasks and  $C_A$ 's.

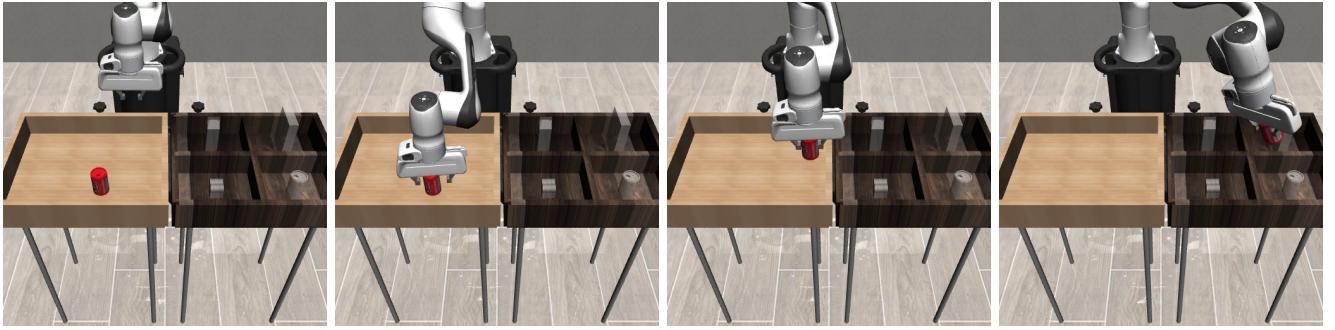


Fig. 17: Snapshot of PPC task rollout

### B. Concepts

Due to the nature of proprioceptive sensors not using image inputs, we do not define  $C_I$  as images like traditional methods. We use the inputs vectors as  $C_I$  the vector input of the particular concept would be taken from the input and rest of the dimension would be randomly selected for example  $C_I = 'object'$  would be the vector input of object(size 10-14) + random vectors(size 9) to make up the total dimension 19. Random concept were random vectors of size 19-23.

### C. Additional Results

For this experiment we test BaTCAVe on the final linear layer which is just after the MLP layer in the architecture Fig 5(a)

## III. EXPERIMENT 3: CUBE LIFTING WITH VISION-LANGUAGE INPUTS

### A. Concepts

The model used in experiment 3 takes in both image and language input, We keep one input constant to test concepts of the other input. Random concepts were samples from ImageNet.

- 1) Image concept: We choose concepts from the input image we blur out the rest of the concept in the input to represent a concept as shown in Fig 23. BaTCAVe shows high variance but consistent score across all  $C_A$  when  $C_I = 'Image'$ .
- 2) Language concept: We use proper instructions, gibberish instructions and just verbs as language concepts. Table III shows the chosen 3 concepts for the experiment. Fig 24 shows BaTCAVe tested across all  $C_A$  in LC task with language used as  $C_I$ .

### B. Model details

**Architecture:** The network outputs a 360-dimensional action vector via a final linear layer, mapping the robot's projected movements over the next 36 timesteps. This output includes the first 108 nodes for end-effector position ( $x, y, z$ ), the subsequent 108 for orientation (roll, pitch, yaw), and the final 36 for gripper.

The main components are Visual Encoder, Visual Narrower, Task ID Encoder, Controllers

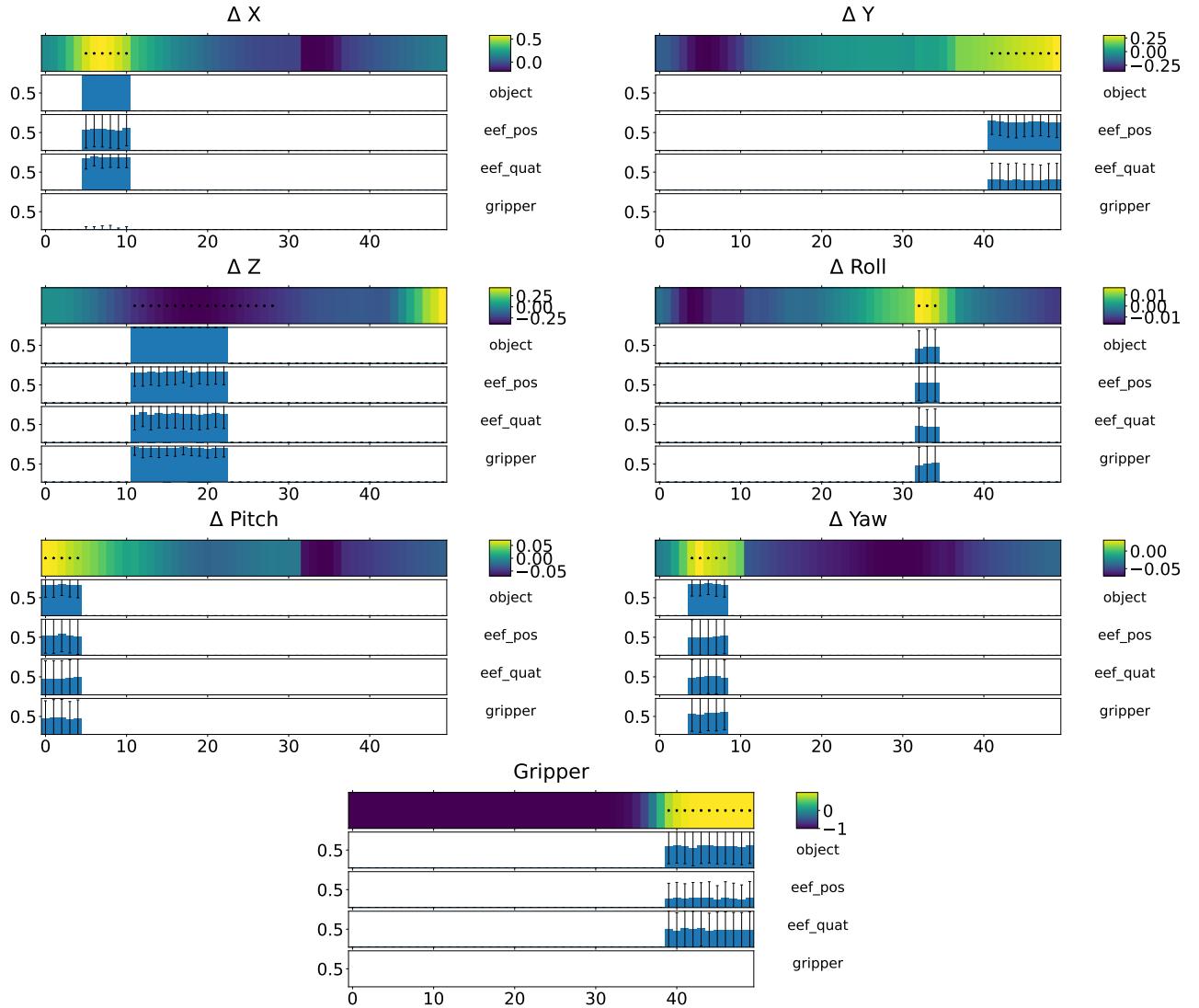


Fig. 18: BeTCAVe across multiple  $C_A$  on task PPC

- 1) Visual Encoder: This part of the model uses a ResNet (Residual Network) architecture for visual encoding.
- 2) Visual Narrower: A linear transformation that reduces the feature dimension from the output of the ResNet (512 features) down to a lower-dimensional space (256 features).
- 3) Task ID Encoder(CLIP): Vision transformer for encoding task IDs based on both textual
- 4) Controllers
  - a) XYZ Controller: Controls the position in 3D space.
  - b) RPY Controller: Controls rotation around the roll, pitch, and yaw axes.
  - c) Grip Controller: Manages the actions related to opening and closing a robotic gripper.

**Training:** The Model is trained based on a behavior cloning (BC) policy tailored to handle dual inputs: high-resolution images (224x224x3) from the camera and verbal instructions given to the robot. We train the model for 100,000 epochs on 300 demonstrations with a batch size of 64 using a Huber loss function and Adam optimizer.

#### IV. EXPERIMENT 4: VISION-BASED AUTONOMOUS DRIVING

##### A. Model details

**Architecture:** The policy model architecture comprises a sequence of convolutional layers followed by fully connected layers. The input to the model is an RGB image, representing the current state of the environment. After the convolutional layers, a flattening operation is applied, transforming the 3D feature maps into a 1D feature vector. This vector serves as the input to the fully connected layers. The flattened vector is fed into a dense (fully connected) layer comprising 512 units. This layer integrates the features extracted by the convolutional layers to form a high-level representation of the input.

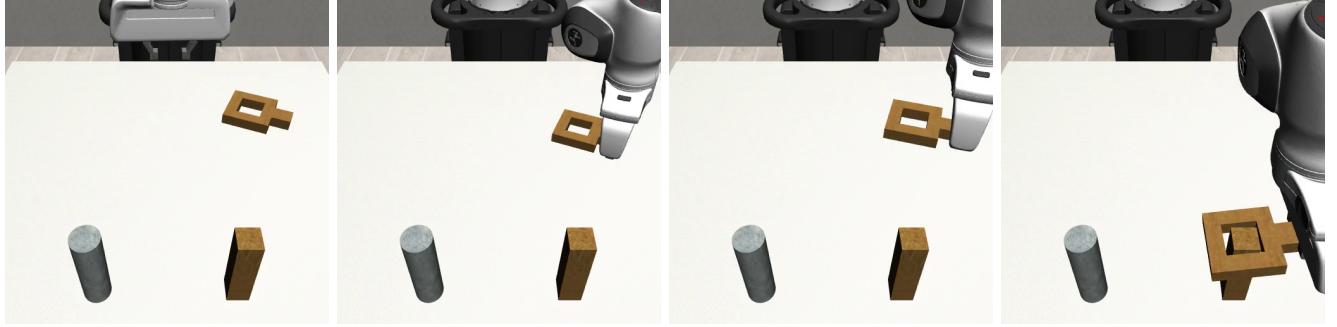


Fig. 19: Snapshot of NA task rollout

TABLE II: BaTCAVe scores across different tasks, quantifying the relevance of each action concept to the task, along with uncertainty estimates. Each score measures the impact of input concepts such as the object’s features, end-effector position (eef\_pos), end-effector orientation (eef\_quat in quaternion format), and gripper on task performance. (\*Not Applicable)

Task	Action Concept	object	eef_pos	eef_quat	gripper
Pick & place	$\Delta X (\uparrow)$	$1 \pm 0.0$	$0.54 \pm 0.49$	$0.96 \pm 0.16$	$0.09 \pm 0.09$
	$\Delta Y (\uparrow)$	$0.0 \pm 0.0$	$0.7 \pm 0.41$	$0.24 \pm 0.43$	$0.0 \pm 0.0$
	$\Delta Z (\uparrow)$	$0.9 \pm 0.17$	$0.85 \pm 0.35$	$0.77 \pm 0.41$	$0.91 \pm 0.27$
	$\Delta \text{Roll} (\uparrow)$	$0.44 \pm 0.49$	$0.53 \pm 0.49$	$0.43 \pm 0.49$	$0.49 \pm 0.49$
	$\Delta \text{Pitch} (\uparrow)$	$0.86 \pm 0.33$	$0.55 \pm 0.49$	$0.45 \pm 0.49$	$0.46 \pm 0.49$
	$\Delta \text{Yaw} (\uparrow)$	$0.84 \pm 0.35$	$0.51 \pm 0.49$	$0.50 \pm 0.49$	$0.58 \pm 0.49$
	Gripper ( $\uparrow$ )	$0.1 \pm 0.0$	$0.64 \pm 0.46$	$0.56 \pm 0.48$	N.A*
	$\Delta X (\uparrow)$	$0.64 \pm 0.47$	$0.59 \pm 0.49$	$0.44 \pm 0.49$	$0.87 \pm 0.32$
Nut assembly	$\Delta Y (\uparrow)$	$0.87 \pm 0.32$	$0.59 \pm 0.48$	$0.46 \pm 0.49$	$0.37 \pm 0.48$
	$\Delta Z (\uparrow)$	$0.005 \pm 0.07$	$0.54 \pm 0.49$	$0.51 \pm 0.49$	$0.85 \pm 0.35$
	$\Delta \text{Roll} (\uparrow)$	$0.52 \pm 0.49$	$0.48 \pm 0.49$	$0.48 \pm 0.49$	$0.45 \pm 0.49$
	$\Delta \text{Pitch} (\uparrow)$	$0.66 \pm 0.46$	$0.51 \pm 0.49$	$0.45 \pm 0.49$	$0.56 \pm 0.49$
	$\Delta \text{Yaw} (\uparrow)$	$0.58 \pm 0.49$	$0.44 \pm 0.49$	$0.45 \pm 0.49$	$0.52 \pm 0.49$
	Gripper ( $\uparrow$ )	$0.1 \pm 0.0$	$0.88 \pm 0.31$	$0.53 \pm 0.47$	N.A*
	$\Delta X (\uparrow)$	$0.84 \pm 0.31$	$0.42 \pm 0.40$	$0.42 \pm 0.49$	$0.42 \pm 0.49$
	$\Delta Y (\uparrow)$	$0.85 \pm 0.32$	$0.99 \pm 0.07$	$0.56 \pm 0.49$	$0.49 \pm 0.40$
Lift	$\Delta Z (\uparrow)$	$0.98 \pm 0.10$	$0.99 \pm 0.04$	$0.42 \pm 0.49$	$0.39 \pm 0.48$
	$\Delta \text{Roll} (\uparrow)$	$0.46 \pm 0.49$	$0.41 \pm 0.49$	$0.48 \pm 0.48$	$0.51 \pm 0.49$
	$\Delta \text{Pitch} (\uparrow)$	$0.54 \pm 0.49$	$0.28 \pm 0.73$	$0.43 \pm 0.47$	$0.48 \pm 0.49$
	$\Delta \text{Yaw} (\uparrow)$	$0.73 \pm 0.39$	$0.28 \pm 0.45$	$0.47 \pm 0.49$	$0.48 \pm 0.49$
	Gripper ( $\uparrow$ )	$0.95 \pm 0.18$	$0.004 \pm 0.06$	$0.31 \pm 0.39$	N.A*

TABLE III: Language concepts

Index	Standard Instructions	Gibberish	Verb
1	Lift the box	skdfj 12asj 5893 2467*	Lift
2	Grab the box	fjdkl c33kd 3940 8175	Grab
3	Take the box	qpwie b99fs 1295 375476	Take
4	Move the box	zxcvb n66gh 5421 983613	Move
5	Collect the box	plmok u55wr 7864 2319	Collect
6	Retrieve the box	akyse l44qs 6572 048756	Retrieve
7	Hoist the box	bvgfr t22vp 3187 7695*(	Hoist
8	Handle the box	nmjqw o11lm 9538 65	Handle
9	Carry the box	xswed k88ht 2409 186428	Carry
10	Raise the box	ecrvt f77yr 4812 690391	Raise

**Training:** The policy network was trained on PPO algorithm using Stable Baselines3 library. The parameters set for training included a learning rate of 0.0003, a rollout of 2048 steps per update, and a batch size of 64. Discount factor (gamma) was set at 0.99 with a Generalized Advantage Estimation (GAE) lambda of 0.95, which helps in balancing bias and variance. The policy clipping range was set to 0.2, and advantages were normalized to stabilize the training. Additionally, the value function coefficient was set at 0.5, and the maximum gradient norm was capped at 0.5 to prevent exploding gradients.

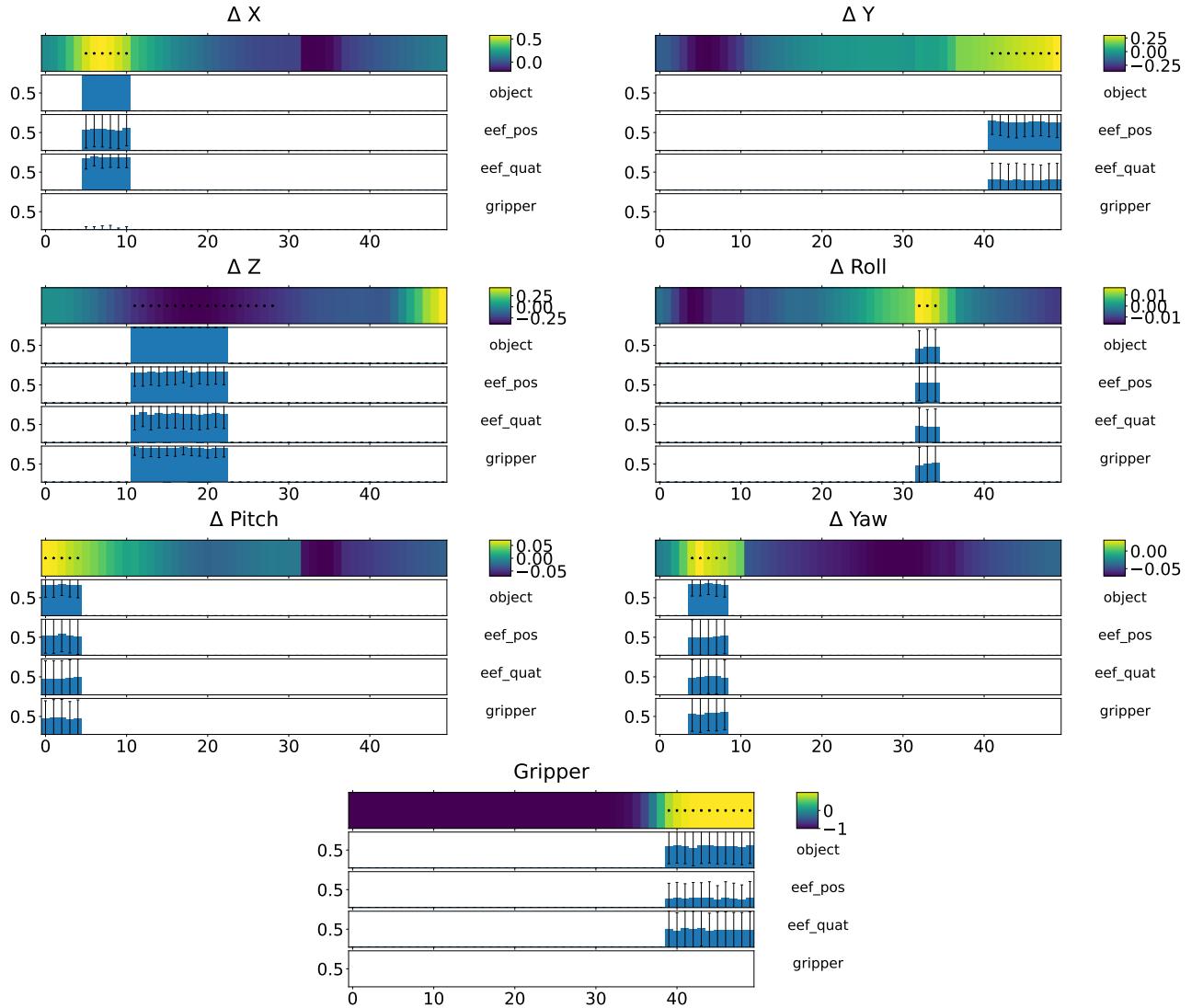


Fig. 20: BeTCAVe across multiple  $C_A$  on task NA

### B. Concepts

As highlighted in section V as one of the limitations, when the concepts are small, they might not provide a strong enough signal to distinguish from a random concept class. For instance, Cone Concept B and Random Concept in Fig. 25 are largely similar.

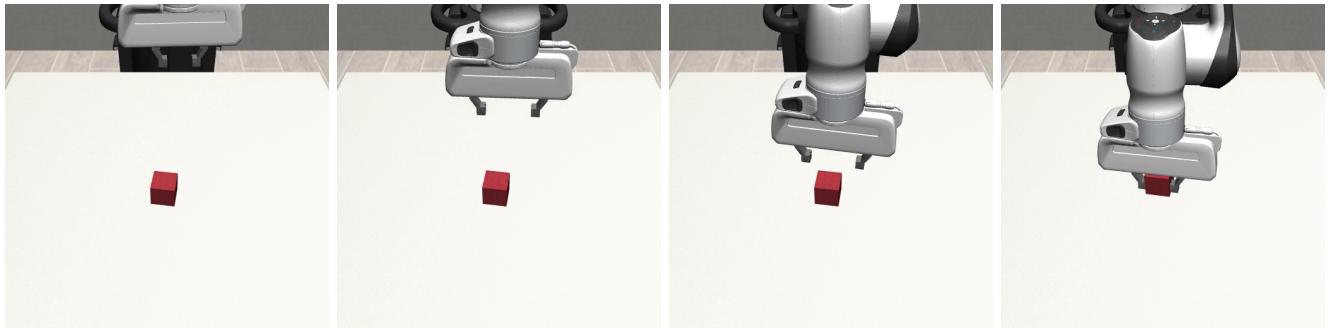


Fig. 21: Snapshot of LC task rollout

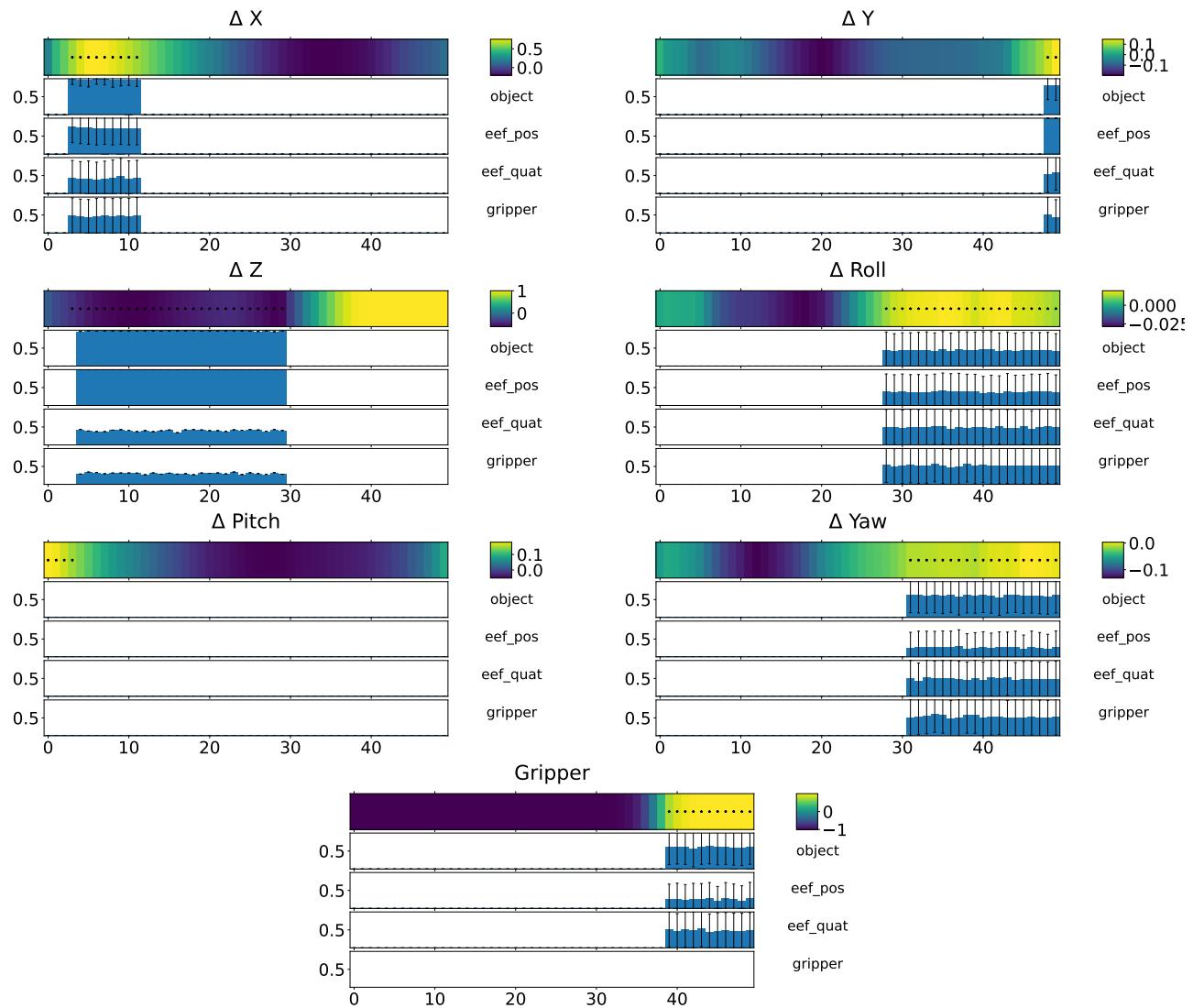


Fig. 22: BeTCAVe across multiple  $C_A$  on task LC



Fig. 23: a) Input b) Red c) Table d) End effector

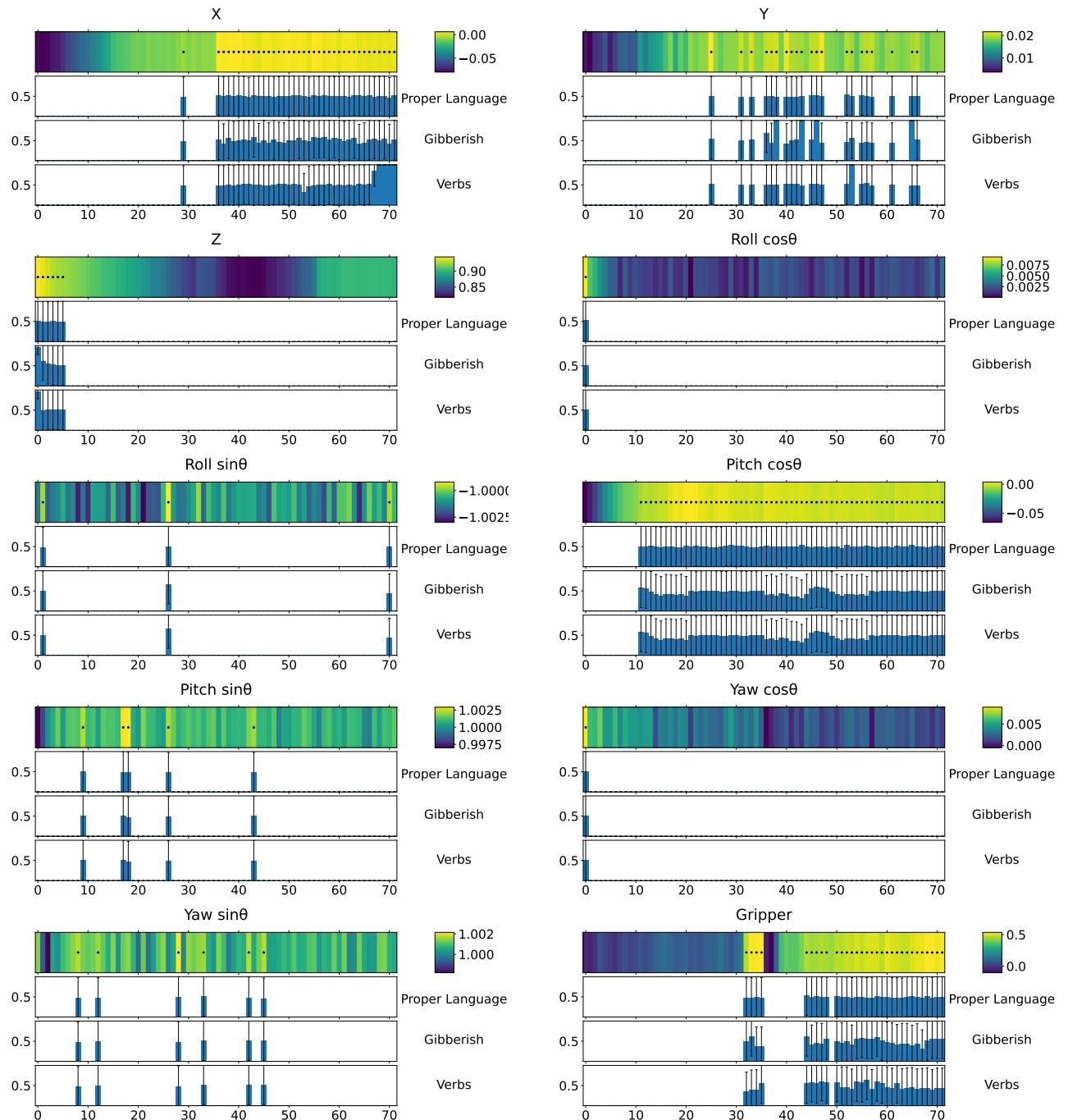


Fig. 24: BeTCAVe across multiple  $C_A$  on task LC with language concept

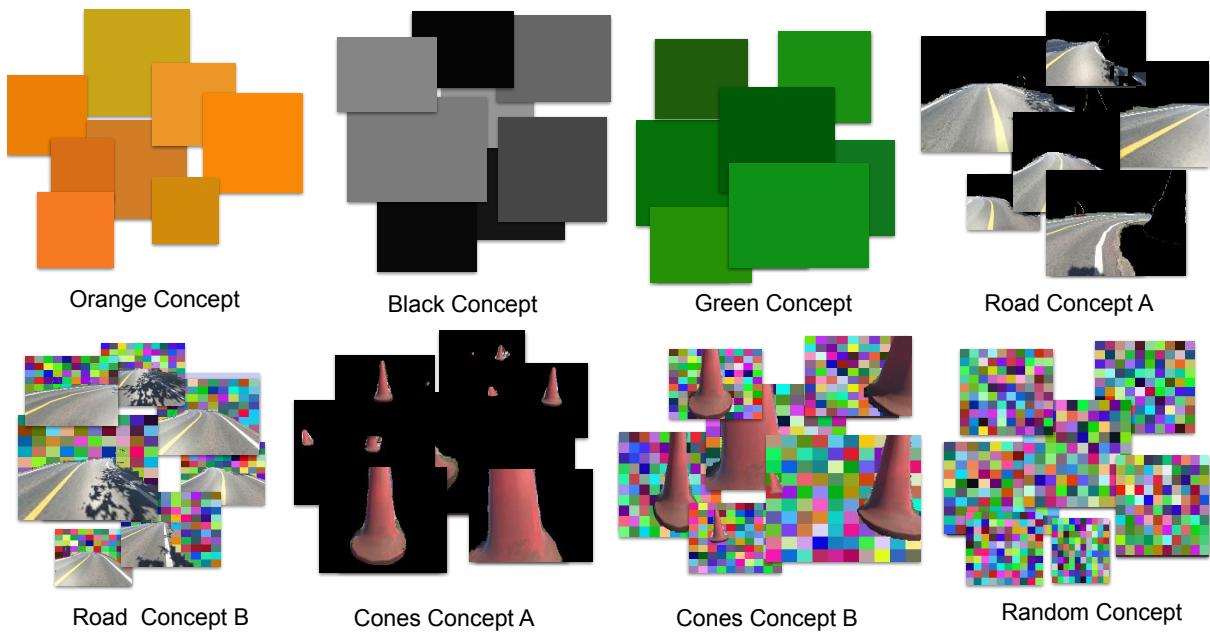


Fig. 25: Different concepts used in BaTCAVe for vision-based autonomous driving task.