Trustworthy Conceptual Explanations for Neural Networks in Robot Decision-Making

Anonymous authors

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 for arbitrary concepts using variational inference on a concept classifier within an explainable manifold. 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.**

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. Although we do not fully understand these black box DNNs, due to their remarkable accuracy in certain test cases we cannot simply discard them. Waiting for them to achieve superior accuracy in all cases is impractical as well, because failures are inevitable regardless of our efforts to build robust models. Instead, we advocate for developing new methods to explain how they work. While there are many ways to build inherently interpretable models [1, 2], the increasing trend of developing larger models and fine-tuning them rather than training from scratch makes embedding interpretability or approximating with simpler models challenging. Therefore, instead of focusing on white-box or gray-box models, this paper focuses on post-hoc explainable machine learning (ML) techniques.

While there are many post-hoc explainable techniques [1, 3, 4], 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 auditing. Therefore, to make explanations human-centric, we consider *concepts*, defined as high-level attributes that help humans understand the black box [5]. As an example, the concept of stripes explains why a DNN would classify an image as a zebra. Unlike feature attribution methods [1, 6, 3, 7, 4], concepts do not need to be a contiguous collection of pixels.

There could be instances where explanations can be wrong or there could 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 metrics, we consider a posterior distribution over concept activation vectors, which, due to the non-exponential-family likelihood, is approximated using variational inference.

II. Related Work

Explainability in robotics: Majority of real-world robots are designed to be interpretable by construction. They are typically equipped with white-box planning [8] and control [9] algorithms. This interpretability can be provided in two different forms: parameters of an inherently interpretable model can be estimated using a data-driven method [10], or an inherently interpretable model can be used for the core decision-making component of an algorithm [11, 12]. Unlike these methods, when inherently black-box neural networks are used for both perception and decision-making, whether in a modular or an end-to-end fashion, we have to develop techniques to probe and explain the network.

Explainable AI: The goal of explainable AI (XAI) is developing techniques to explain black-box models. Certain techniques achieve this by isolately testing various components of an input through perturbations [3] or component removal [6], while other models achieve this through local approximation of the global decision boundary [6]. When explaining neural networks, XAI techniques might use gradients [4], node level information [13], or layer-wide information [5]. For image inputs, explanations can be provided in the form of highlighting certain parts of an image [4], providing importance scores to semantically meaningful segments or a cluster of pixels [1, 6], or samples [2, 5]. Typical XAI techniques do not provide estimations of uncertainty about their explanations, making them difficult to trust in high-stakes applications such as robotics. Authors of [14] have also highlighted the importance in trustworthiness of explanation in robotics settings.

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 [5, 15]. While concept-based explanations have been applied in domains such as biomedical imaging [16], they are still applied to typical discrete image classification settings. In

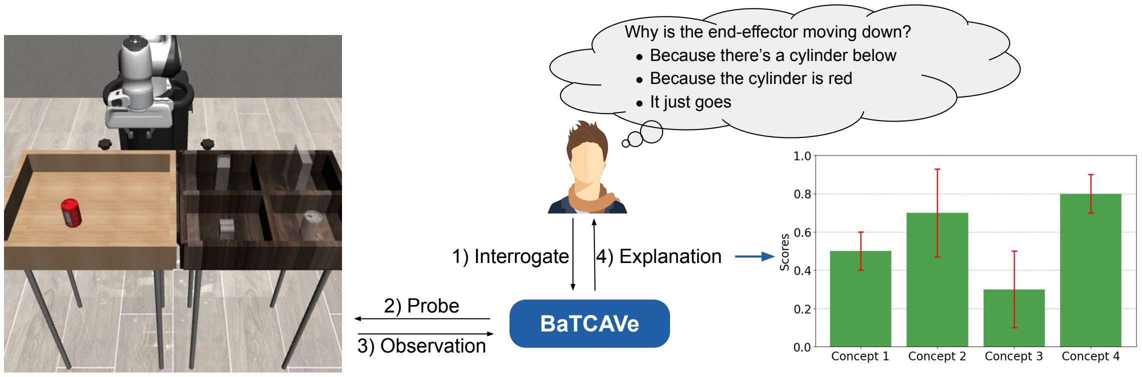


Fig. 1: The explainability pipeline. The user obtains a score with uncertainty for each "concept."

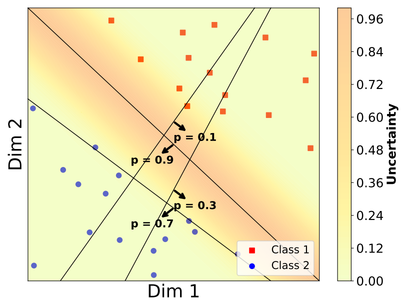


Fig. 2: The activation space, $f_{-l}(\mathbf{x})$. Arrows indicate four samples from $\mathbf{v}$

this paper, we propose a method to provide explanations for decisions and control commands using concepts.

III. Method

By building a relationship between activations and human cognition, humans can obtain explanations on how the neural networks work. To build this relationship, concept activation vectors (CAVs) can be used. They measure how sensitive a classifier's output is to some activation's in the direction of a concept of interest [5]. However, this notion cannot be used in robotic decision-making since robot actions contain continuous control signals, for which CAVs are not defined. Also, they do not provide uncertainty estimates of explanations, as illustrated in Fig. 1. We propose the following definitions before presenting the method which offers a metric to assess a concept's explanation of robot decisions and its epistemic uncertainty.

Action Concepts: An action concept, C_A, is a set of conditions that defines a subset of the output space a user is interested in analyzing.

Input Concepts: An input concept, C_I, is a high-level, human interpretable attribute that the user believes is important to explain an action concept. These can be textures, colors, sizes, distances, directions, shapes, objects, etc.

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 Concept Activation Vectors (BaTCAVe)

Given a collection of input concepts for an action concept, we now derive a score to measure how well each concept explains the action concept. As a simple example, this score can tell if the skin tones black or white (input concepts) affect the breaking action (action concept) of a DNN used in an autonomous vehicle. To formally define, let the activations at the l^{th} layer be $f_{-l}(\cdot)$. If these activations are passed through the network after the l^{th} layer, $f_{+l}(\cdot)$, then we obtain the output, $\mathbf{y}$. In other words, $\mathbf{y} = f(\mathbf{x}) = f_{+l}(f_{-l}(\mathbf{x}))$ for input $\mathbf{x}$. We can obtain the sensitivity of neural network decisions, conditioned by C_A, w.r.t. layer activations as $\frac{\partial f^{C_A}(\mathbf{x})}{\partial f_{-l}(\mathbf{x})}$. As shown in Fig. 2, this sensitivity in the direction of a random variable, V, with its realizations, $\mathbf{v}$, is given by the stochastic directional derivative,

$$\int \left(\lim_{h\to 0} \frac{f_{+l}^{C_A}\left(f_{-l}(\mathbf{x}+h\mathbf{v})\right) - f_{+l}^{C_A}\left(f_{-l}(\mathbf{x})\right)}{h} \cdot p(\mathbf{v}) \right) \mathrm{d}\mathbf{v} = \int \frac{\partial f^{C_A}(\mathbf{x})}{\partial f_{-l}(\mathbf{x})} \cdot \mathbf{v} \cdot p(\mathbf{v})\, \mathrm{d}\mathbf{v}\,. \tag{1}$$

If we define the random variable, V, in such a way that represents C_I, then the sensitivity is measured with respect to the input concepts. The probability density function of V is estimated using an external dataset containing inputs from C_I. To this end, similar to [5], we collect $X^{C_I} = \{(\mathbf{x}_m^{C_I}, z^+)\}_{m=1}^{M}$ and a different set of images $X^{C_I^-} = \{(\mathbf{x}_m^{C_I^-}, z^-)\}_{m=1}^{M}$ 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. More details on BaTCAVe formalization is in Appendix I.

IV. Experiments

We consider three types of algorithms for learning robot decision-making: binary supervised learning, behavioral cloning (BC), and proximal policy optimization (PPO). Since candidate concepts can be human defined [5], extracted from

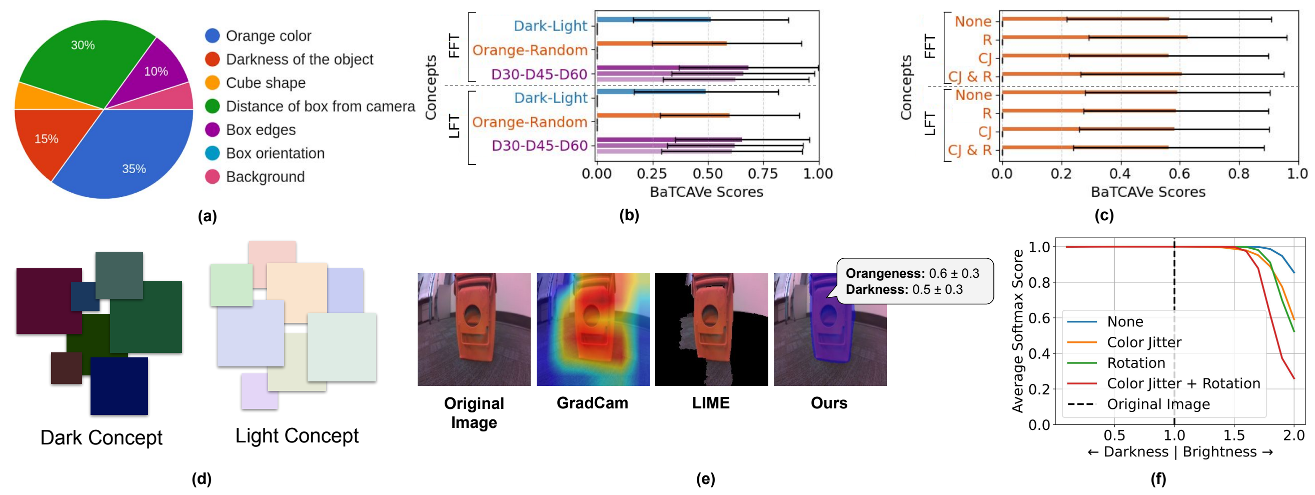


Fig. 3: (a) Shows the distribution of input concepts C_I selected by the participants for object avoidance task. (b) and (c) shows the effects of fine-tuning and data augmentation, respectively. The higher the score, the better the explanation is. (d) shows samples of dark and light concepts. (e) shows while common XAI methods such as GradCam [4] and LIME [1] can highlight the orange box, they do not reveal what attributes (i.e., input concepts) 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. (f) highlights the change in confidence over modifying darkness factor in input with models trained with different data augmentation (C-Modification).

dataset [15, 17, 18], or automatically generated through vision-language models [19], we assume they are given. The experimental setup and results are detailed in the Appendix II. We have summarize our findings below.

A. Experiment 1: Mobile Robot Navigation Using Vision

With the objective of avoiding orange obstacles, we fine-tuned an AlexNet model with images of an orange box, until we achieve a validation accuracy of 100%. Details about the experiment setup can be found in Appendix II-A.

As shown in Fig. 3(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. As shown in Fig. 3(f), similar to C-deletion in XAI literature [15, 17, 18], by gradually varying the brightness of the orange object we verified that the darkness is an important concept. Additionally, based on Fig. 3, BaTCAVe made the following explanations: noitemsep, topsep=0pt

- The BaTCAVe score is proportional to the concept of "distance between the robot and obstacle," verifying that the DNN has learned the distance, as an engineer would expect.
- 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 3(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.

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

In this setup, we obtain notably high BaTCAVe score with a low uncertainty, corroborating that accurate object information from proprioceptive sensors, unlike in vision-based settings, helps with precisely performing the task. Table I shows 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 4b, explains that only the object pose and EEF position measurements matters when the EEF is moving down.

C. Experiment 3: Lifting a Cube with Vision-Language Inputs

In this experiment we evaluate image and language concepts (C_I = [images={cube, gripper, table}, language={proper language commands, gibberish, verbs}]) for robot-decision making. 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, as shown in Fig. 5a. More details about the experiment and results in Appendix II-C.

D. Experiment 4: Autonomous Driving with Vision

In this experiment we analyze how various input concepts affect steering decisions (C_A = steering angle > 2). Fig. 6 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

TABLE I: BaTCAVe scores across different tasks along with uncertainty estimates. Each score measures the impact of C_I. For complete table refer Table II in Appendix.

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

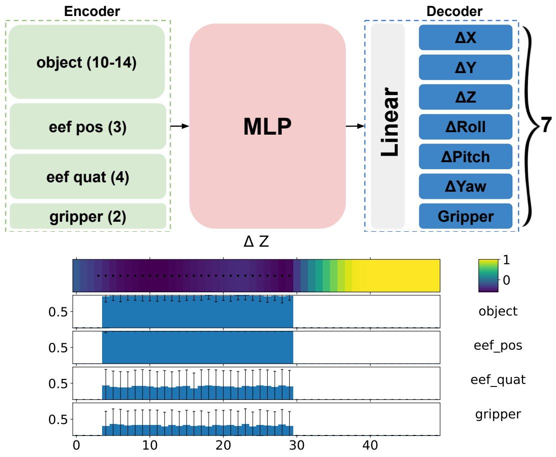


Fig. 4: (a) Architecture (b) Task : Lift, $C_A = \Delta Z$. The colorbar indicates the Δ actions and the dots on them indicate the actions we consider by following the rule C_A =top 25% of moving down. The three bar plots indicate the BaTCAVe scores with their uncertainties for three input concepts EEF position, quaternion, and gripper status. (c) Lift-cube task (more details in Appendix II-B).

MLP layer ($\bar{s} = 0.822$), we found that errors do not originate in the CNN (image encoder), indicating that the policy needs to be trained more.

V. Conclusion

We proposed a task-agnostic explainable robotics technique. We demonstrated how our method can be used to explain various robot actions and 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.

Social Impact Statement: Our work introduces BaTCAVe, a novel technique that provides human-understandable explanations for robot decisions using high-level concepts, along with uncertainty scores to assess trustworthiness. By enhancing transparency, this approach empowers engineers to debug systems more effectively and enables policymakers to audit AI-driven technologies, fostering safer and more reliable deployment of robots in critical real-world applications.

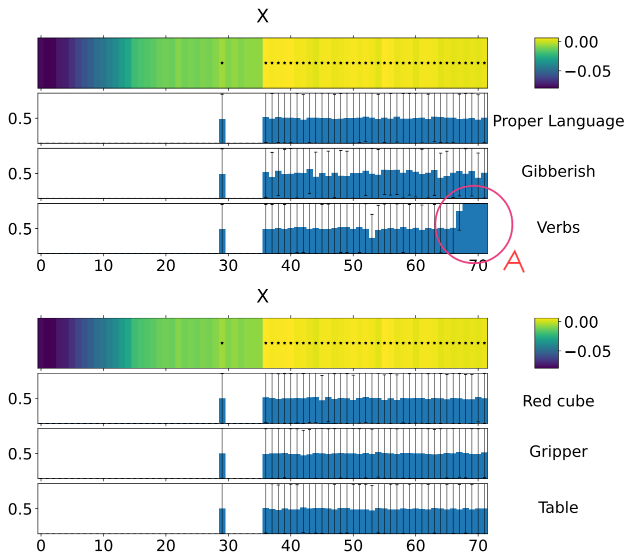


Fig. 5: (a) Indicates verbs in a language command matter most when it is lifting. (b) Indicates the images concepts are highly uncertain.

BaTCAVe ratio
Black/ Orange = 1.05
Black / Green = 1.03
Green / Orange = 2.41
Random / Cones = 1.93

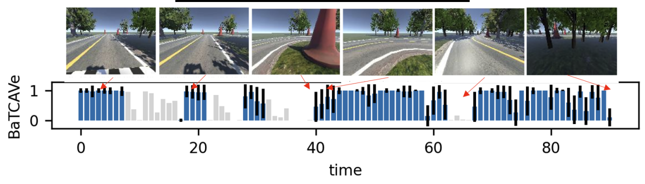


Fig. 6: (a) The relative importance of concepts indicates that the agent relies on the black concept (∼ road) than the orange concept (∼ cones). (b) Temporal change of scores.

References

[1] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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*, pages 1135–1144, 2016.

[2] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[3] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[4] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[5] 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.

[6] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[8] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *arXiv preprint arXiv:1709.10256*, 2017.

[9] Dimitri Bertsekas. *Reinforcement learning and optimal control*, volume 1. Athena Scientific, 2019.

[10] Raunak P Bhattacharyya, Ransalu Senanayake, Kyle Brown, and Mykel J Kochenderfer. Online parameter estimation for human driver behavior prediction. In *2020 American Control Conference (ACC)*, pages 301–306. IEEE, 2020.

[11] Stephanie Milani, Zhicheng Zhang, Nicholay Topin, Zheyuan Ryan Shi, Charles Kamhoua, Evangelos E Papalexakis, and Fei Fang. Interpretable multi-agent reinforcement learning with decision-tree policies. In *Explainable Agency in Artificial Intelligence*, pages 86–120. CRC Press, 2024.

[12] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. A survey of explainable reinforcement learning. *arXiv preprint arXiv:2202.08434*, 2022.

[13] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[14] Lindsay Sanneman and Julie A Shah. Trust considerations for explainable robots: A human factors perspective. *arXiv preprint arXiv:2005.05940*, 2020.

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

[16] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020.

[17] 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.

[18] 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.

[19] Aditya Taparia, Som Sagar, and Ransalu Senanayake. Explainable concept generation through vision-language preference learning. *arXiv preprint arXiv:2408.13438*, 2024.

[20] Tommi S. Jaakkola and Michael I. Jordan. A variational approach to Bayesian logistic regression models and their extensions. In David Madigan and Padhraic Smyth, editors, *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, volume R1 of *Proceedings of Machine Learning Research*, pages 283–294. PMLR, 04–07 Jan 1997.

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

[22] George J Klir and Arthur Ramer. Uncertainty in the dempster-shafer theory: a critical re-examination. *International Journal of General System*, 18(2):155–166, 1990.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[24] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[25] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei,

Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Appendix

I. Bayesian Testing with Concept Activation Vectors (BaTCAVe)

To estimate the vector distribution, $\mathbf{v}$, that separates the two classes in the activation space (Fig. 2), we apply the Bayes theorem,

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

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. 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. However, since the true posterior is not know, instead of minimizing the KL divergence, we maximize an evidence lower bound (ELBO),

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

Following the locally linear approximation of the posterior in [20], we learn $q(\mathbf{v})$ in an expectation-maximization-fashion. With the approximate posterior distribution, $q(\mathbf{v})$, estimated using variational inference, we can then use (Eq 1) on a collection of test inputs $X^{\text{test}} = \{\mathbf{x}_p^{\text{test}}\}_{p=1}^{P}$, to compute a score s^{C_A, C_I, C_I^-},

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

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

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

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^{test}. Similarly, the empirical standard deviation reflects the epistemic nature [21] of explanations—how much the model knows that its explanation can be wrong.

A. Interpreting the Trustworthiness of Explanations

To assess the trustworthiness of explanations, relying on 1) the held-out test accuracy of the estimation in eq. (2) and 2) the epistemic uncertainty of the score in eq. (4), we consider (Fig. 7):

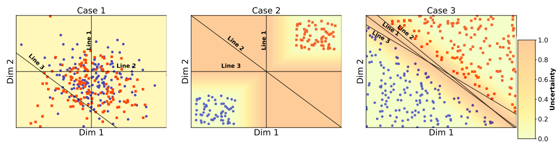


Fig. 7: Data with two classes (red and blue) are represented in the activation space. If the uncertainty is high (case 2 vs. 3), then we can sample many lines (i.e., many explanations). Though many lines can be sampled from case 1 as well, since the accuracy is low, the explanations can be trusted.

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 is not capable of differentiating lack of information from conflicting information [22].

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 of the highest level of trustworthiness.

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 threshould 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.

II. Experimental Setup

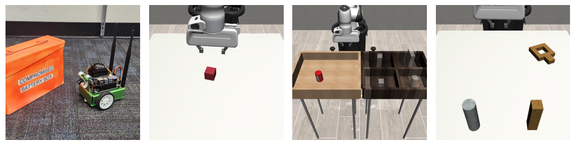


Fig. 8: a) JetBot b) Lift cube c) Pick-and-place d) Nut assembly

A. Experiment 1: Mobile Robot Navigation Using Vision

We used a JetBot with a NVIDIA Jetson Nano (Fig 8a). It uses a pretrained AlexNet [23] 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 $C_I = \{\text{orangeness, darkness, distance}\}$. Fig 9 shows the snapshots of JetBot rollout.

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. 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 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.

1) Model details: **Architecture**: AlexNet [23] 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.

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.

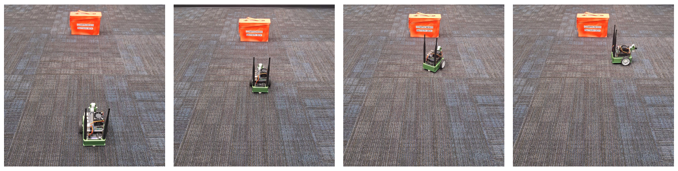


Fig. 9: Snapshot of JetBot rollout

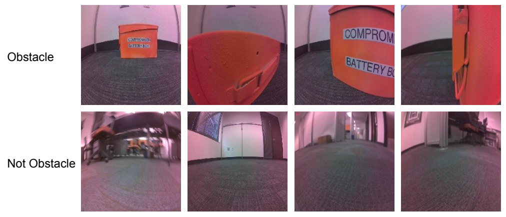


Fig. 10: Orange box Obstacle and Not obstacle dataset sample

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.

2) 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. These concepts are based on the response we received from the human survey described in later sub-section.

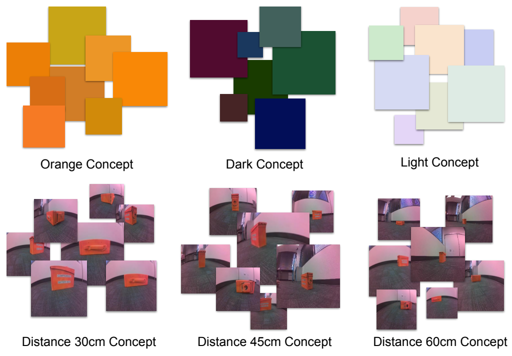


Fig. 11: Different concepts sets used in BaTCAVe for Experiment 1.

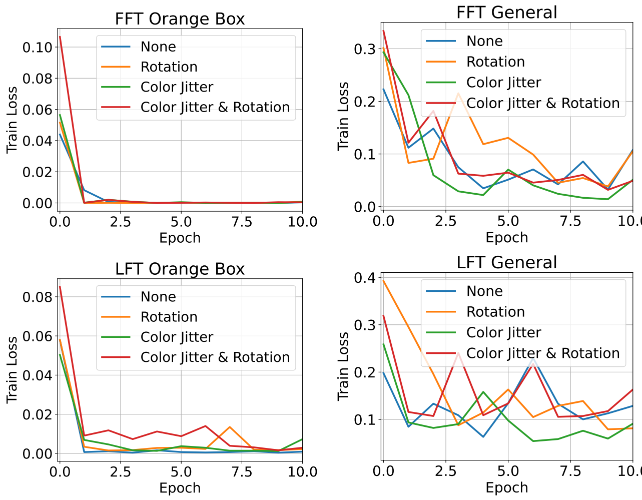


Fig. 12: Training loss of FFT (left) and LFT (right) over orange obstacle and general obstacle dataset and different augmentations.

3) Additional Results: 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.

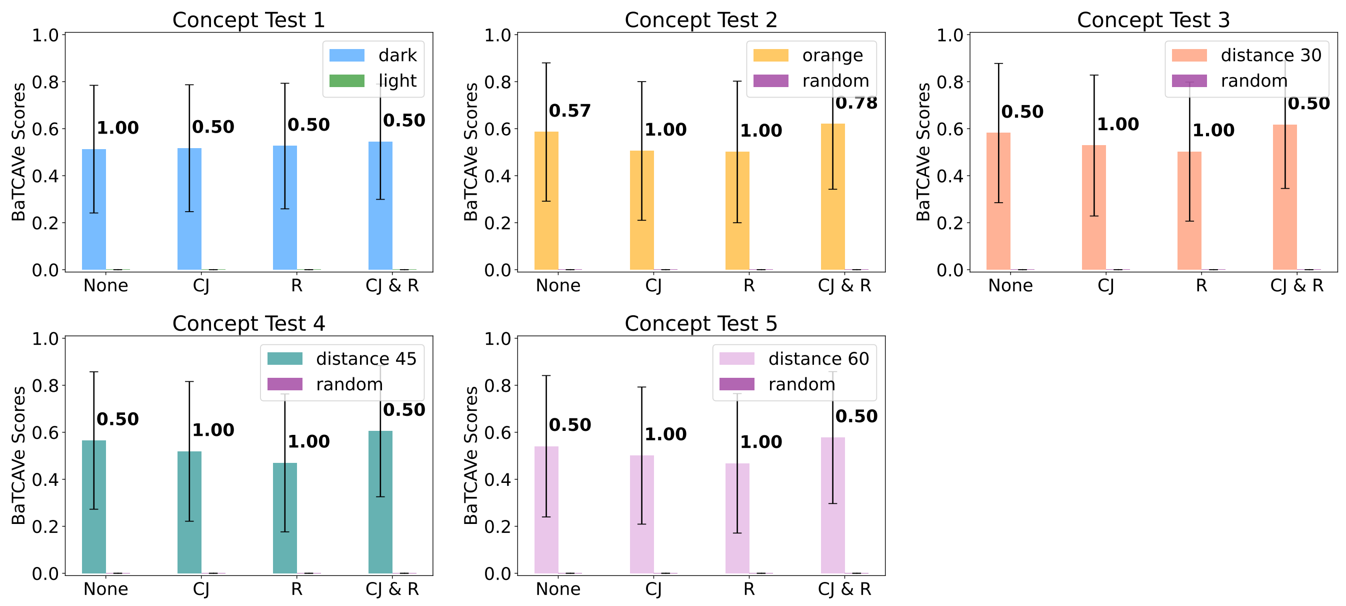


Fig. 13: Different concepts test on FFT model trained on general dataset over different augmentations. (BaTCAVe's classifier accuracy is shown near the bar graph for each augmentation.)

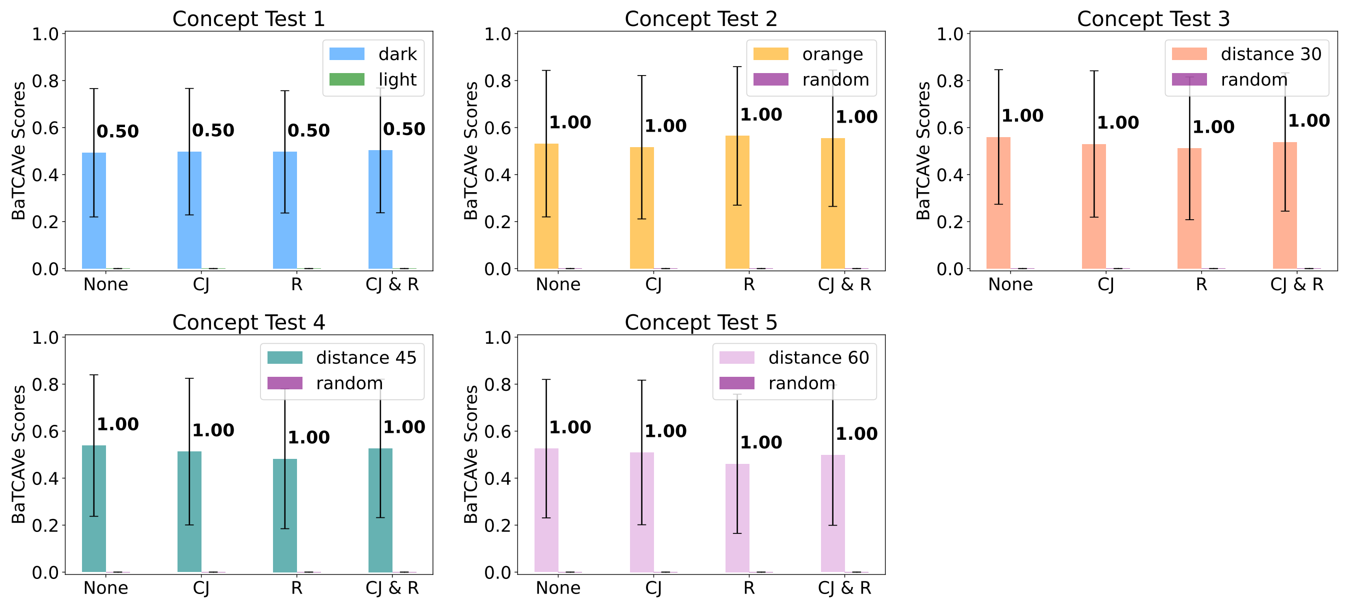
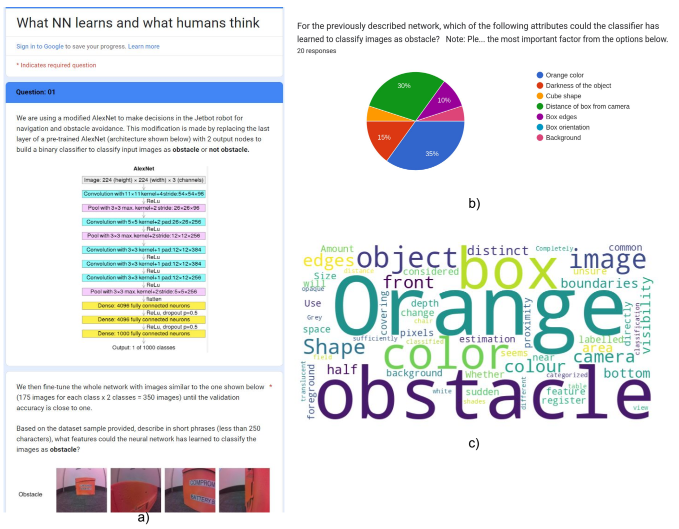


Fig. 14: Different concepts test on LFT model trained on general dataset over different augmentations. (BaTCAVe's classifier accuracy is shown near the bar graph for each augmentation.)

4) Human survey: 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. As shown in Fig 15(a), we first 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 fined-tuned to distinguish orange from the rest. 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. These results helped us to formulate candidate concepts while testing the models.

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

We use a Panda, a 6-DOF robotic arm with a gripper, to complete the three tasks shown in Fig 8(b,c,d) on RoboSuite [24]. The setup includes various proprioceptive sensors to monitor the arm's movements and positions. By using the



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) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max. kernel+2 stride: 12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max. kernel+2 stride:5×5×256
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 has learned to classify images as obstacle? Note: Ple... the most important factor from the options below.
20 responses

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

DNN shown in Fig 4 and the training procedure, we developed an agent based on behavioral cloning [25]. Robot's actions are Δ differences. Our objective is to explain which concepts of the inputs, C_I= 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 = top 75% of each Δ action.

1) Model details: **Architecture**: The model architecture, depicted in Fig 4(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.

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 [25] on task can,lift and sqare. 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 task:

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.
2) Nut Assembly (NA): This task requires the robot to pick up a hollow object and accurately pick its handle and fit it in the object which fits the hollow area.
3) Lift Cube (LC): In the LC task, the robot's goal is to lift a cube from the table.

Fig. 16 shows training loss over epoch on all the different tasks. Table II shows BeTCAVe scores across all tasks and C_A's.

2) 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 perticular concept would be taken from the input and rest of the dimesnion 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.

3) Additional Results: For this experiment we test BaTCAVe on the final linear layer which is just after the MLP layer in the architecture Fig 4(a). The results across all the different tasks are as follows:

1) Pick Place Can (PPC): Fig 17 shows the state across different timesteps and Fig 18 shows BaTCAVe performed across multiple C_A on PPC.
2) Nut Assembly (NA): Fig 19 shows the state across different timesteps and Fig 20 shows BaTCAVe performed across multiple C_A on NA.
3) Lift Cube (LC): Fig 21 shows the state across different timesteps and Fig 22 shows BaTCAVe performed across multiple C_A on LC.

C. Experiment 3: Lifting a Cube with Vision-Language Inputs

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.

1) 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.

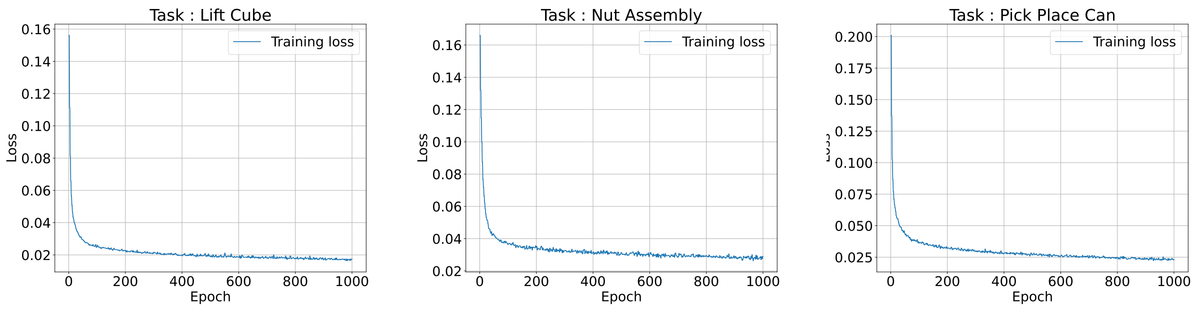


Fig. 16: Training loss of model across tasks

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	Δ X (↑)	1 ± 0.0	0.54 ± 0.49	0.96 ± 0.16	0.09 ± 0.09
	Δ Y (↑)	0.0 ± 0.0	0.7 ± 0.41	0.24 ± 0.43	0.0 ± 0.0
	Δ Z (↑)	0.9 ± 0.17	0.85 ± 0.35	0.77 ± 0.41	0.91 ± 0.27
	Δ Roll (↑)	0.44 ± 0.49	0.53 ± 0.49	0.43 ± 0.49	0.49 ± 0.49
	Δ Pitch (↑)	0.86 ± 0.33	0.55 ± 0.49	0.45 ± 0.49	0.46 ± 0.49
	Δ Yaw (↑)	0.84 ± 0.35	0.51 ± 0.49	0.50 ± 0.49	0.58 ± 0.49
	Gripper (↑)	0.1 ± 0.0	0.64 ± 0.46	0.56 ± 0.48	N.A*
Nut assembly	Δ X (↑)	0.64 ± 0.47	0.59 ± 0.49	0.44 ± 0.49	0.87 ± 0.32
	Δ Y (↑)	0.87 ± 0.32	0.59 ± 0.48	0.46 ± 0.49	0.37 ± 0.48
	Δ Z (↑)	0.005 ± 0.07	0.54 ± 0.49	0.51 ± 0.49	0.85 ± 0.35
	Δ Roll (↑)	0.52 ± 0.49	0.48 ± 0.49	0.48 ± 0.49	0.45 ± 0.49
	Δ Pitch (↑)	0.66 ± 0.46	0.51 ± 0.49	0.45 ± 0.49	0.56 ± 0.49
	Δ Yaw (↑)	0.58 ± 0.49	0.44 ± 0.49	0.45 ± 0.49	0.52 ± 0.49
	Gripper (↑)	0.1 ± 0.0	0.88 ± 0.31	0.53 ± 0.47	N.A*
Lift	Δ X (↑)	0.84 ± 0.31	0.42 ± 0.40	0.42 ± 0.49	0.42 ± 0.49
	Δ Y (↑)	0.85 ± 0.32	0.99 ± 0.07	0.56 ± 0.49	0.49 ± 0.40
	Δ Z (↑)	0.98 ± 0.10	0.99 ± 0.04	0.42 ± 0.49	0.39 ± 0.48
	Δ Roll (↑)	0.46 ± 0.49	0.41 ± 0.49	0.48 ± 0.48	0.51 ± 0.49
	Δ Pitch (↑)	0.54 ± 0.49	0.28 ± 0.73	0.43 ± 0.47	0.48 ± 0.49
	Δ Yaw (↑)	0.73 ± 0.39	0.28 ± 0.45	0.47 ± 0.49	0.48 ± 0.49
	Gripper (↑)	0.95 ± 0.18	0.004 ± 0.06	0.31 ± 0.39	N.A*

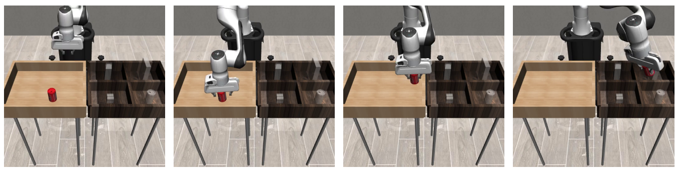


Fig. 17: Snapshot of PPC task rollout

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.

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 ($224 \times 224 \times 3$) 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.

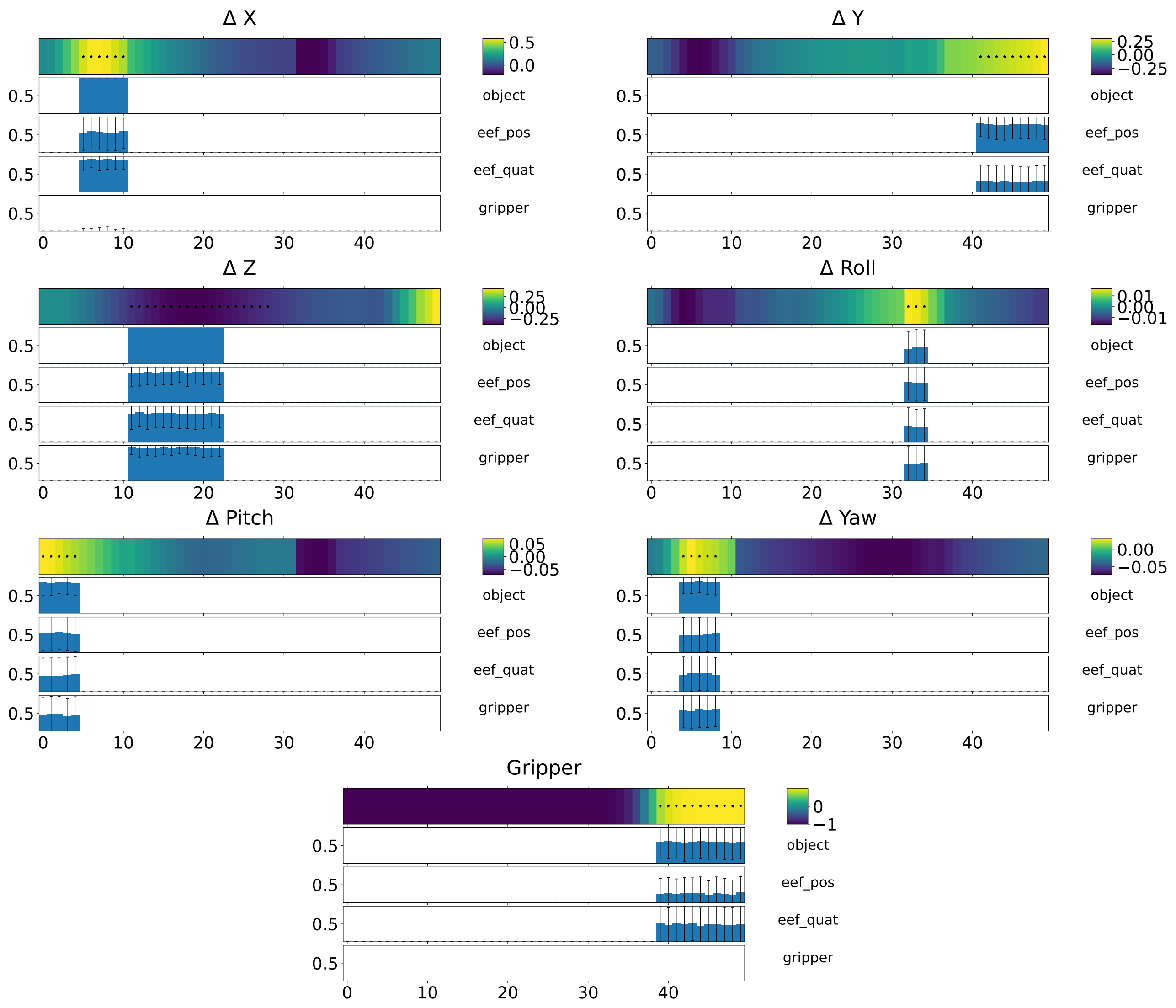


Fig. 18: BeTCAVe across multiple C_A on task PPC

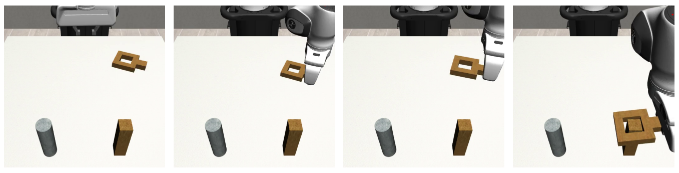


Fig. 19: Snapshot of NA task rollout

2) 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 eperiment. Fig 25 shows BaTCAVe tested across all C_A in LC task with language used as C_I.

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

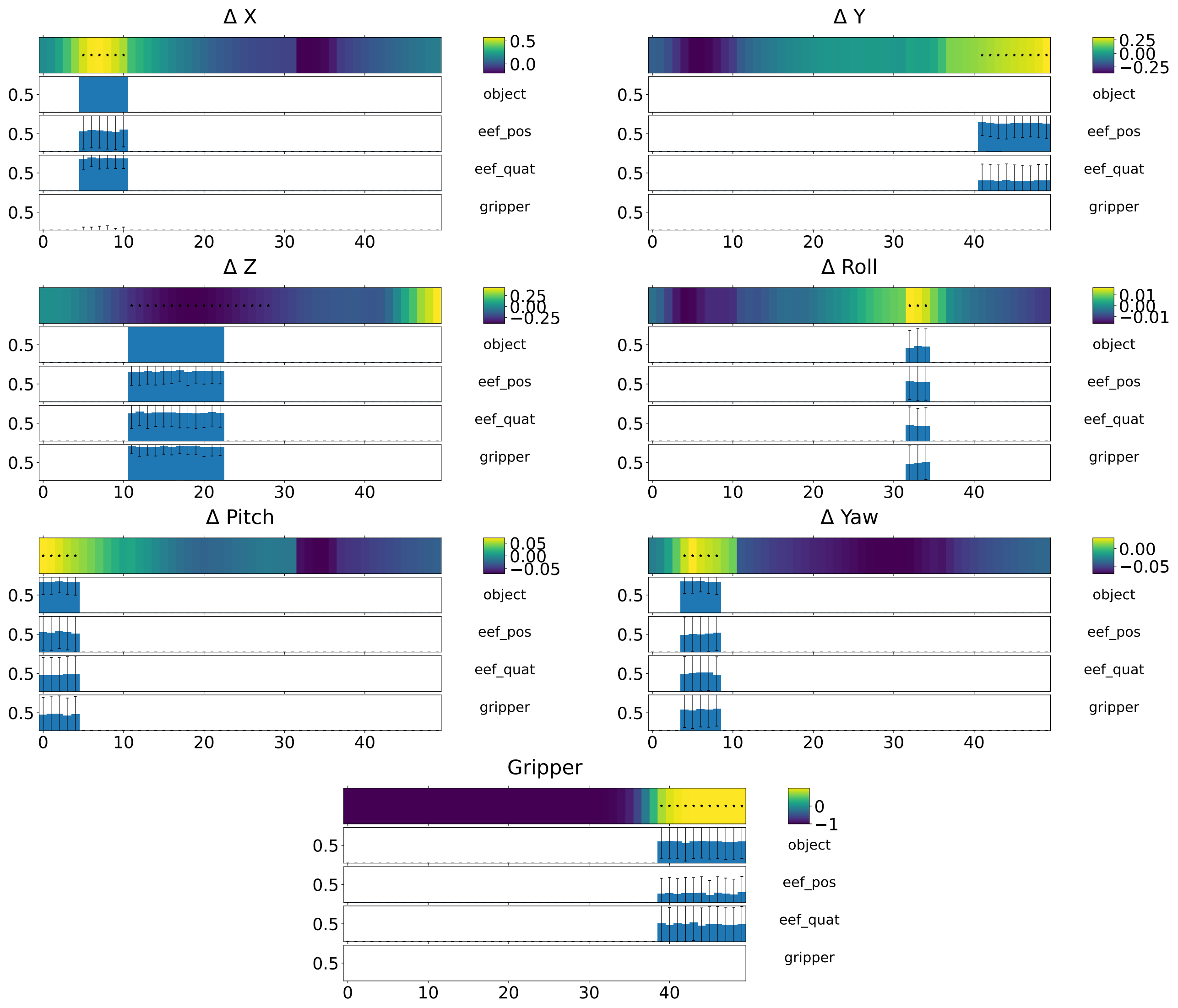


Fig. 20: BeTCAVe across multiple C_A on task NA

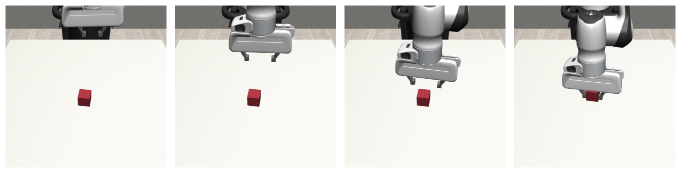


Fig. 21: Snapshot of LC task rollout

D. *Experiment 4: Autonomous Driving with Vision*

To simulate autonomous driving, we trained an end-to-end deep reinforcement learning policy using proximal policy optimization (PPO) [26] to steer and throttle a vehicle on Donkey Simulator. The network is a CNN followed by an MLP, trained end-to-end.

1) 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.

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.

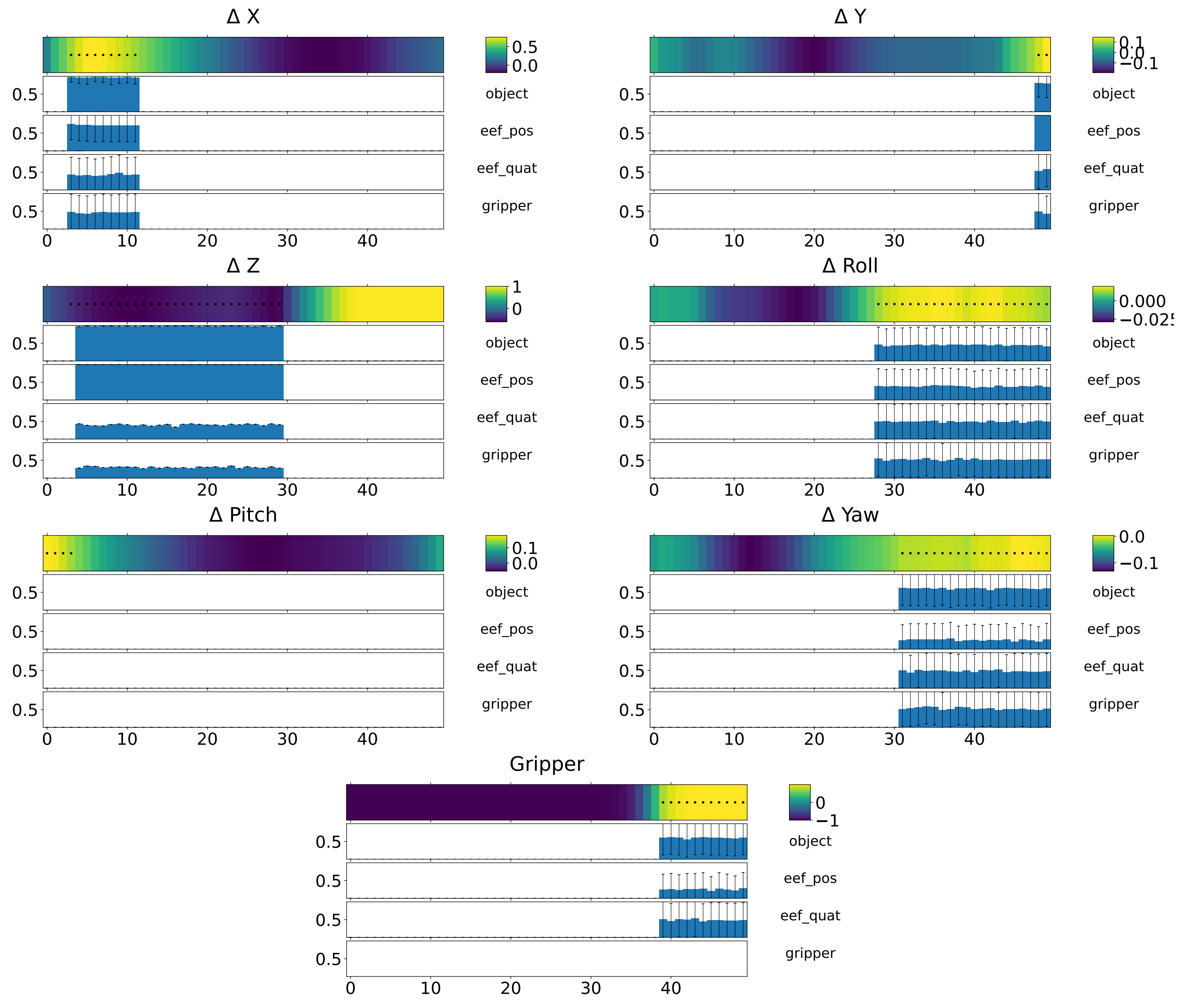


Fig. 22: BeTCAVe across multiple C_A on task LC

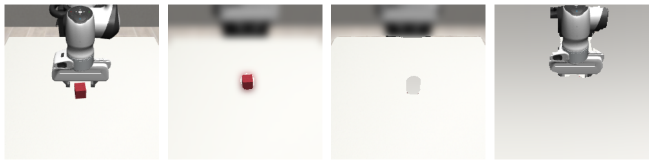


Fig. 23: a) Input b) Red c) Table d) End effector

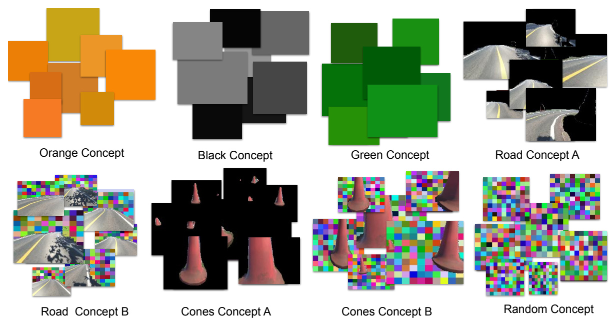


Fig. 24: Different concepts used in BaTCAVe

2) Concepts: One of the limitations of BaTCAVe is 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. 24 are largely similar.

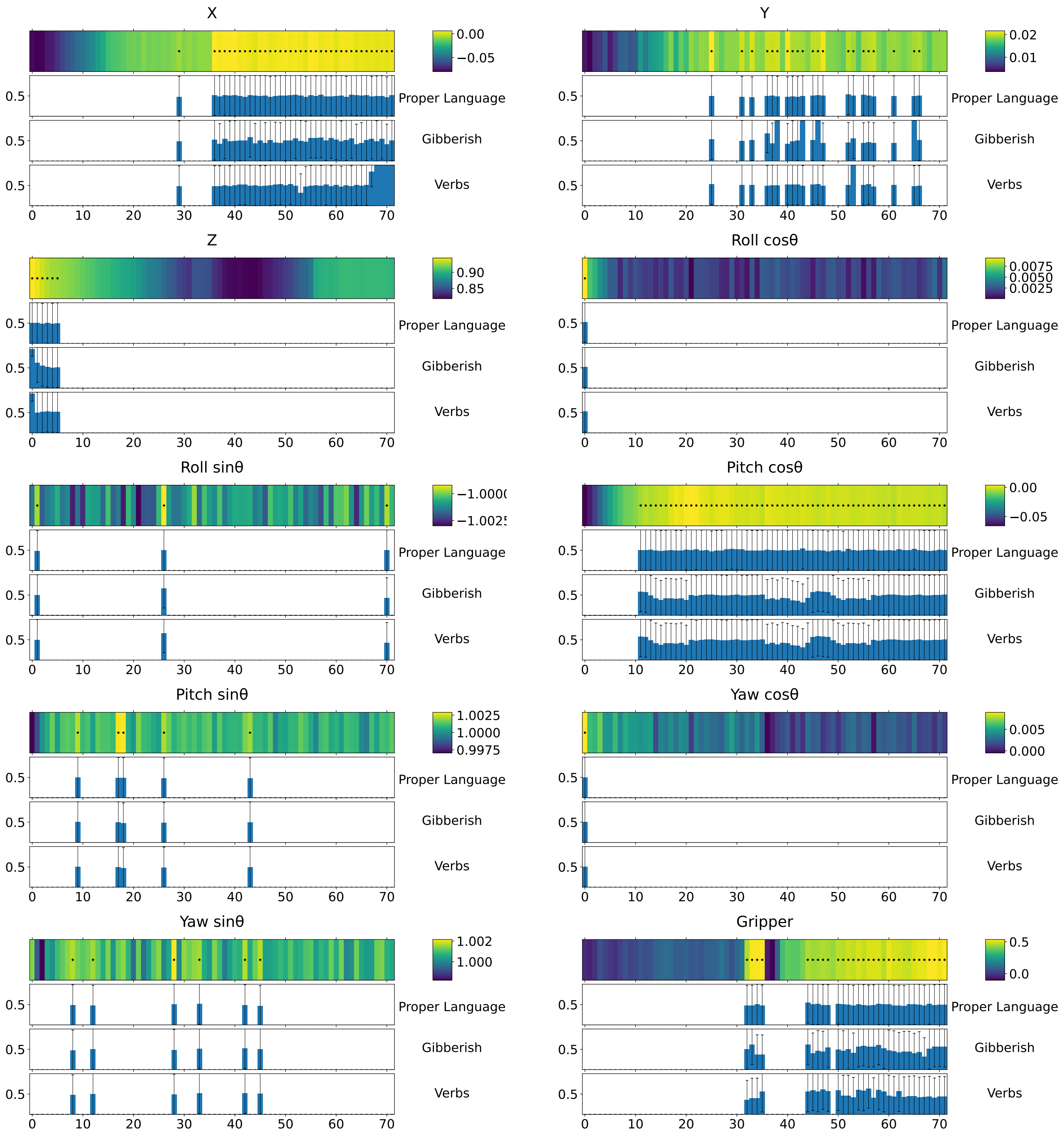


Fig. 25: BeTCAVe across multiple C_A on task LC with language concept