Run-time Observation Interventions Make Vision-Language-Action Models More Visually Robust

Asher J. Hancock[1], Allen Z. Ren[1], and Anirudha Majumdar[1]

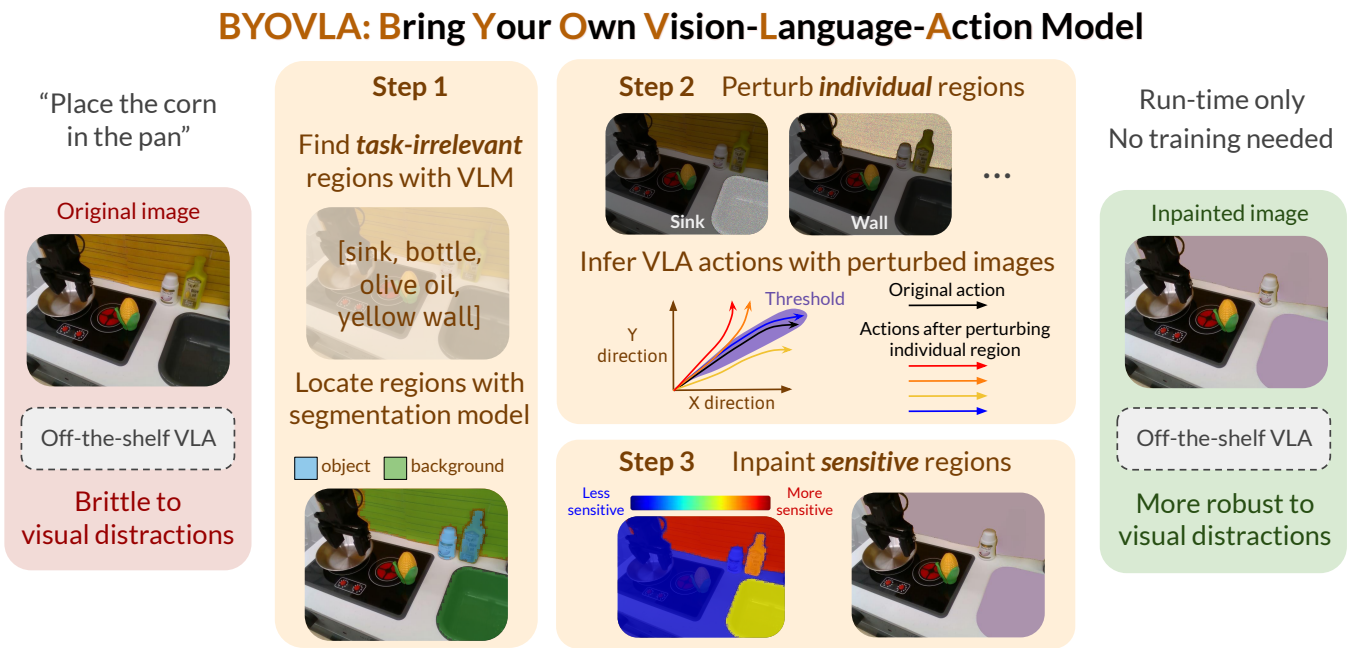

Fig. 1: We introduce BYO*VLA*: a simple and lightweight run-time intervention scheme for improving the performance of an arbitrary VLA model in the presence of task-irrelevant distractions. Our method identifies task-irrelevant regions in the visual observation and minimally modifies regions that the model is sensitive to in order to reduce sensitivity to distractors.

***Abstract*— Vision-language-action (VLA) models trained on large-scale internet data and robot demonstrations have the potential to serve as generalist robot policies. However, despite their large-scale training, VLAs are often brittle to task-irrelevant visual details such as distractor objects or background colors. We introduce *Bring Your Own VLA* (BYO*VLA*): a run-time intervention scheme that (1) dynamically identifies regions of the input image that the model is sensitive to, and (2) minimally alters *task-irrelevant* regions to reduce the model's sensitivity using automated image editing tools. Our approach is compatible with any off the shelf VLA without model fine-tuning or access to the model's weights. Hardware experiments on language-instructed manipulation tasks demonstrate that BYO*VLA* enables state-of-the-art VLA models to nearly retain their nominal performance in the presence of distractor objects and backgrounds, which otherwise degrade task success rates by up to 60%. Website with additional information, videos, and code: https://aasherh.github.io/byovla/.**

I. Introduction

A longstanding goal in robotics is to develop *generalist* robot policies that can be instructed on the fly to perform tasks in diverse environments. Recently, vision-language-action (VLA) models trained with a combination of large-scale internet data and robot demonstrations have shown promise towards such generalization [1, 2, 3, 4]. These models leverage their internet-scale training to perform a broad range of visuomotor control tasks when prompted via natural language.

However, while existing VLAs show broad *task* generalization, they fall short of their promise as generalist policies in terms of variations in *environments*. Due to the complexity of real-world scenarios and the lack of robotic data at scale, state-of-the-art VLAs are brittle against marginal variations in the environments they were trained on. In particular, prior work [1, 2, 3, 5] and our experiments (Sec. IV) have shown a lack of *visual* generalization; a small number of distractor objects or a mere change of background color, which leave the inherent task difficulty invariant, can *drastically lower* the task success rates of VLAs.

While further scaling up data can potentially mitigate such performance drops, the effort required to collect such data and the computational resources required to fine-tune large VLAs (often with billions of parameters) is a strong deterrent. *Can we design a lightweight and model-agnostic tool that does not alter the model weights, yet still improves robustness of VLAs to task-irrelevant objects and backgrounds?*

*This work was partially supported by the NSF CAREER Award [#2044149] and the Office of Naval Research [N00014-23-1-2148]. Asher Hancock was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2146755.

[1]Dept. of Mechanical & Aerospace Engineering, Princeton University. Contact: ajhancock@princeton.edu.

Contributions. To this end, we propose *Bring Your Own VLA* (BYO*VLA*): a run-time intervention scheme that improves visual generalization of off the shelf VLAs by minimally altering regions in the VLA's visual inputs in order to reduce sensitivity against visual distractors. The key idea is to identify (at run-time) which regions of the visual input the model is sensitive to using a *visual sensitivity probe* that perturbs different segments of the visual input. BYO*VLA* queries a vision-language model (VLM) to identify which regions in the environment are task-irrelevant and alters a region using automated image editing tools (e.g., inpainting a distractor object) if the region is task-irrelevant *and* the VLA is sensitive to it (Fig. 1).

BYO*VLA* can be applied to any VLA model without fine-tuning or access to the model's weights. Across multiple language-instructed manipulation tasks and varying distractor objects and backgrounds, BYO*VLA* improves task success rates by $20-60\%$ compared to the original VLA, while also significantly improving performance relative to baselines that perform run-time interventions (1) without accounting for the model's visual sensitivity or (2) assessing sensitivity via prior image attribution methods (e.g., GradCAM [6]).

II. Related Work

A. Vision-Language-Action (VLA) models

Building upon progress in foundation models for language and vision [7], recent years have seen the rise of generalist vision-language-action (VLA) models [1, 2, 3, 5, 8] which show early promise in performing diverse tasks when prompted via natural language. This success has been enabled by a combination of existing internet data and large-scale efforts towards collecting human demonstration datasets such as Open X-Embodiment [4] and DROID [9].

Nevertheless, the complexity of real-world scenarios still overwhelms the amount of data available, and state-of-the-art generalist VLAs are often brittle against minor visual changes to the scene, such as the introduction of task-irrelevant objects or differing backgrounds. For instance, [5] demonstrates that Octo [3] — a recently proposed VLA trained on Open X-Embodiment data — has its task success rate dropped from 60% to 29% in visual generalization tasks consisting of object distractions and unseen object appearances or backgrounds.

B. Improving policy robustness to visual distractors

There have been multiple lines of research for ameliorating the effect of visual distractors on a policy's performance. A straightforward and widely used method is to apply large-scale domain randomization to the visual observation [10, 11], often in the form of random noise or simple image manipulation such as cropping and translating. Recently, automated image editing tools (e.g., inpainting) have been used to generate diverse and more realistic backgrounds and object textures for data augmentation [12, 13, 14, 15]. These methods all apply such randomization *during training*, whereas BYO*VLA* operates at run-time only and does not alter the model weights.

Another technique is to simply *mask out* the background and possibly the irrelevant objects in the scene, either by learning a masking module end-to-end [16] or using the segmentation object mask [17, 18, 19]. However, simple masking can make unrealistic observations which the model is subsequently trained on. A recent work [20] uses a VLM to determine the relevant objects in the scene based on the task instruction, but again masks out *task-irrelevant regions* and requires training with both the original and edited images. BYO*VLA* does not require model re-training, and applies *selective* masking based on model sensitivity. As current inpainting tools are imperfect, such minimal edits help mitigate artifacts potentially generated by the image editing process, keeping the transformed observations relatively realistic (as Fig. 1 shows).

To our knowledge, [21] is the only other work that *solely* performs run-time interventions of the camera observation for robot manipulation policies, but does not address visual distractions. Rather they focus on ensuring that *task-relevant* objects are within the training distribution by inpainting novel ones with those seen during training.

Other training strategies for improving model robustness include creating bottlenecks in the attention mechanism [22, 23] of the policy architecture to train the policy to selectively focus on objects [24, 25]. Similar attention effects can also be achieved using information bottlenecks [26, 27] or bisimulation-based state abstractions [28, 29] for learning visual representations that only encode task-relevant information. Again these methods all require altering the training pipeline and are thus not compatible with VLAs off the shelf, unlike BYO*VLA*.

C. Determining the task-relevant elements in the scene

As discussed above, previous work has investigated using VLMs [20] or learning an end-to-end module [16] to determine the task-irrelevant elements in the scene. BYO*VLA* also leverages the rich prior knowledge of VLMs to identify regions of the scene that are irrelevant, but visually manipulates them only if the model is sensitive to them.

Our use of model sensitivity is also related to multiple *attribution* methods — usually used in image classification settings — that seek to determine which part of the image (input) are most responsible for the model's output. Methods like SHAP [30] and LIME [31] determine how each input feature contributes to the output by learning a small model or a few parameters. Gradient-based methods such as GradCAM [6] and SmoothGrad [32] compute how the model output changes as parts of the input observation are perturbed. However, these methods tend to be brittle and unreliable [33, 34]; specifically, the results can be sensitive to implementation details, such as the specific layer of the model network with respect to which the gradient is computed, or they may be entirely incorrect. In contrast, the visual sensitivity probe we introduce in Sec. III *directly* measures changes in action outputs by perturbing different segments of the visual input. As our experiments in Sec. IV show, determining sensitivity using GradCAM *does not* retain the base model's nominal performance in the presence of distractions.

Algorithm 1 Bring Your Own VLA (BYO*VLA*)

Require: VLA model f, observation o_t, language instruction l, threshold τ
$\mathcal{R}_t \leftarrow$ TASK-IRRELEVANT REGIONS(o_t)
$(a_t, \ldots, a_{t+T_a}) \leftarrow f(o_t, l)$
Initialize $\rho_f(o_t) = o_t$
for each region $r \in \mathcal{R}_t$ **do**
  $\tilde{o}_t \leftarrow$ PERTURB REGION(o_t, r)
  $(\tilde{a}_t, \ldots, \tilde{a}_{t+T_a}) \sim f(\tilde{o}_t, l)$
  $\Delta_f(o_t, r) \leftarrow$ Eq. (2) ▷ Calculate visual sensitivity
  if $\Delta_f(o_t, r) \geq \tau$ **then**
    $\rho_f(o_t) \leftarrow$ IMAGE EDITOR($\rho_f(o_t)$, r)
  end if
end for
return $\rho_f(o_t)$

III. METHODOLOGY

A. Problem formulation

Our goal is to improve the performance of a pre-trained VLA operating in environments with task-irrelevant visual distractions. We consider policies $f(o_t, l)$ that take a language instruction l as input in order to perform a visuomotor control task using RGB image observations o_t. In contrast to prior work (Sec. II), we propose a purely *run-time* intervention that does not require any model fine-tuning or access to the model's weights. More formally, our goal is to process the raw observation o_t to produce a new observation $\rho_f(o_t)$ that is then sent as input to the VLA to produce an action chunk (sequence):

$$(a_t, \ldots, a_{t+T_a}) \sim f(\rho_f(o_t), l), \tag{1}$$

where T_a is the action prediction horizon.

The run-time intervention ρ_f manipulates regions of o_t that f is sensitive to but that are *irrelevant* to the task at hand, with the objective of recovering the nominal performance of the VLA *in the absence* of visual distractors. We describe our pipeline for implementing ρ_f in detail below.

B. Bring Your Own VLA

Fig. 1 and Algorithm 1 provide an overview of our approach. Given a language instruction l and an initial observation o_0, we first query a vision-language model (VLM) in order to identify visual regions that are *irrelevant* to the task. At each time-step t during policy execution, we then use a segmentation model to obtain corresponding masks for these irrelevant regions. A key component of our approach is to introduce a *visual sensitivity probe* in order to identify which irrelevant segments the VLA f is sensitive to. The final processed observation $\rho_f(o_t)$ is obtained by manipulating irrelevant regions (e.g., inpainting an object or changing the color of a background region) using automated image editing tools. We describe each of these components below.

Step 1: Localize task-irrelevant objects. Semantic information about an image is readily captured by VLMs [35, 36], which we utilize to determine what regions in the initial image o_0 are task-irrelevant. We run the state-of-the-art GPT4-o model from OpenAI and prompt the model with few-shot exemplars (image observations paired with irrelevant regions in text). The output from GPT4-o is a string of region proposals in o_0 deemed task-irrelevant. The proposals are then provided to a segmentation model, Grounded-SAM2 [37, 38, 39, 40], to localize and partition the regions at the pixel-level at every step of the rollout. Fig. 1 depicts the outputs at each stage of the process. We consider static environments in our experiments, and thus GPT4-o is called once at initialization and the string of region proposals for the grounded segmentation model is held invariant during task execution.

Step 2: Apply visual sensitivity probe. Given a set $\mathcal{R}_t$ of task-irrelevant regions from the VLM and segmentation model, we determine which of these impact the output of the VLA f. We quantify the sensitivity of f to a region $r \in \mathcal{R}_t$ by perturbing the image in that segment to obtain $\tilde{o}_t$ and measuring the change in actions. Specifically, let $(a_t, \ldots, a_{t+T_a}) \sim f(o_t, l)$ denote the predicted action chunk for the original observation o_t. Here, each action in the chunk corresponds to $(x, y, z, \phi, \theta, \psi, g)$: the relative displacements of the end-effector in translation and rotation, along with a gripper open/close state. Let $(\tilde{a}_t, \ldots, \tilde{a}_{t+T_a})$ denote the action chunk for a perturbed observation $\tilde{o}_t$. After applying a single perturbation to region r using a perturbation distribution described below, we sample K action chunks $\{(\tilde{a}^k_t, \ldots, \tilde{a}^k_{t+T_a})\}_{k=1}^K$. Additionally sampling K action chunks $\{(a^k_t, \ldots, a^k_{t+T_a})\}_{k=1}^K$ from the original image, we then compute an average weighted L_2-norm of the difference in actions $\Delta a^k_{t+t'} := a^k_{t+t'} - \tilde{a}^k_{t+t'}$ (where $t' \in \{0, \ldots, T_a\}$):

$$\Delta_f(o_t, r) := \frac{1}{KT_a} \sum_{k=1}^{K} \sum_{t'=0}^{T_a} \sqrt{\langle w \Delta a^k_{t+t'}, \Delta a^k_{t+t'} \rangle}, \tag{2}$$

where $w \in \mathbb{R}^7$ is a user-defined weighting vector. To perturb an image, we consider Gaussian blurring (smoothening) object distractions and adding Gaussian noise to background distractions for reasons explained below.

Determining the sensitivity threshold. If the quantity $\Delta_f(o_t, r)$ in Eq. (2) is greater than a threshold τ for a region r from the segmentation model, we intervene on that region. To determine τ for object distractions, we utilize the first observation from 60 environments in BridgeV2 and apply Gaussian blurring to the task-irrelevant object regions in the image following Step 1 above. Computing Eq. (2) with w as the indicator function for translational components, and then taking the third quartile, we arrive at a value of approximately 0.005m. Since different environments in BridgeV2 are of different physical scales, we adjust the threshold by rolling out a few trials in our kitchenette, arriving at a threshold value for object distractors of $\tau = 0.002\text{m}$.

For background distractions, we use the same value of τ described above but add Gaussian noise to the the RGB channels of the observation, instead of Gaussian blurring, as we find blurring too weak to cause a substantial deviation

in trajectories for these regions. The value for τ described above is used for all experiments.

Step 3: Transform the image. The specific image transformation is dependent upon whether the region is classified as an object or background distraction, which can be determined by the VLM. If the region is an object distraction, a vision model capable of inpainting is called to remove it from the image; in our experiments, we use Inpaint Anything [41].

If the region is a background distraction, the RGB pixels in that region are simply altered such that $\Delta_f(o_t, r) < \tau$. Recall the intuition that we would like the transformed observation to better match the training data. Since the distribution of colors seen during training is hard to specify a priori, we choose a random, neutral color to inpaint the background region with, recalculate Eq. (2) for the inpainted image and inpainted-plus-noised image, and repeat until $\Delta_f(o_t, r)$ is below the threshold. If the sensitivity condition is not reached within N color proposals, we simply choose the RGB pixel coloring that yielded the smallest sensitivity. Throughout all experiments, we set $N = 10$.

IV. Experiments

We evaluate BYO*VLA* with two state-of-the-art open-source VLA models: Octo-Base [3] (93M parameter transformer-based diffusion policy) and OpenVLA [5] (7B parameter transformer-based autoregressive policy). The tasks considered are "put the carrot on yellow plate" and "put the eggplant in the pot," which take place in a toy kitchen environment from the BridgeData V2 dataset [42] and are the representative tasks used for evaluation in [3] and [5].

Environments and hardware setup. In our experiments, we consider object and background environmental distractions. **Object distractions** include items commonly found in a kitchen environment but which are irrelevant for task completion. Importantly, object distractions do not affect the trajectory required by the robot to reach the goal state. In each task, approximately 4-5 object distractions are added to the domain; these objects are selected from the BridgeData V2 catalog of objects and are thus *not adversarial* in nature. **Background distractions** include changes to the appearance of the scene background that are irrelevant to the task. Fig. 2 depicts object and background distractions in our kitchen environment for the task "put the carrot on yellow plate": addition of an "orange fruit" is an object distraction whereas changing the tiling color to yellow is a background distraction. In general, distractions are chosen to be realistic while weakening VLA performance in order to assess the benefits of BYO*VLA*.

Following the hardware experiments from Octo [3] and OpenVLA [5], all policies are evaluated on the Widow X 250S robotic arm in accordance with the setup prescribed by [42]. Camera angles and object/background distractions are held constant during all trials. A threshold of 0.5 for the gripper state is set to determine when to open or close the end-effector. The Widow X is initialized at the same position above the task object; consequently, w is kept as the indicator function for translational components in Eq. (2) for all experiments, the rationale being that we find rotational and gripper commands insignificant to task success if the robot starts from this position. In accordance with [3] and [5], the task object's initial position is varied 1-3cm from a central position between trials. Unless otherwise stated, 15 trials for all baselines are completed.

Run-time. In general, the overhead incurred by BYO*VLA* is reliant on (1) the inference speed of the underlying foundation models and (2) the number of task-irrelevant regions to manipulate. Queries to the VLM (GPT4-o) for determining task-irrelevant objects on average take less than three seconds to complete and cost less than one cent with five few-shot exemplars; we assume static environments, so this query is executed once at the beginning of the episode. Irrespective of inpainting, Octo-Base and OpenVLA run at 13Hz and 6Hz, respectively, on a NVIDIA GeForce RTX 4090 GPU [3, 5], which we used in our experiments. In our object distraction experiments with Octo, Steps 1-3 above, without calling GPT4-o, take roughly 2 seconds to complete. For background distractions, finding a color which the VLA is insensitive to required at most 10 seconds. All time measurements were averaged over 15 trials.

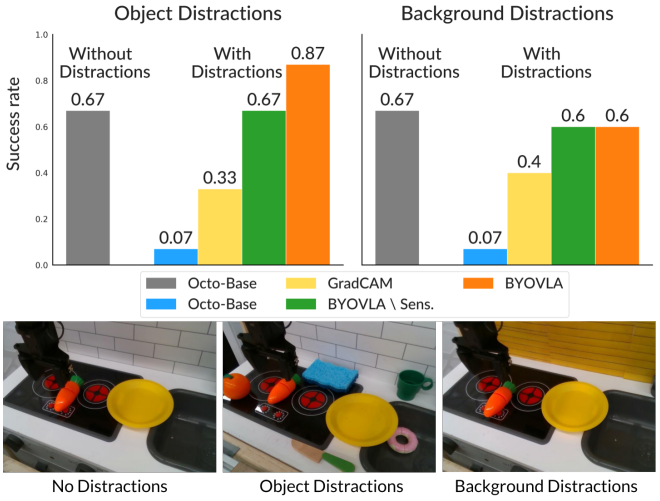

Fig. 2: First row: task success rates for BYO*VLA* with Octo on language instruction "place the carrot on yellow plate." Second row: kitchenette environment from BridgeV2 dataset with and without object and background distractions.

Baselines. We evaluate BYO*VLA* against these baselines: (1) the original VLA policy; (2) BYO*VLA* without the visual sensitivity probe — labeled in Figs. 2 – 4 as BYO*VLA* \Sens. — where we manipulate *all* regions of the image deemed task-irrelevant by the VLM. For our experiments with Octo, we consider (3) a GradCAM-based [6] baseline, where we replace our visual sensitivity probe with GradCAM to attribute what regions in the image are most important for model output and manipulate those regions if they are deemed task-irrelevant by the VLM. In particular, we calculate the cross-attention values and gradients between the image and task tokens, which we then average across the attention heads and task tokens. We perform this operation halfway through the overall transformer

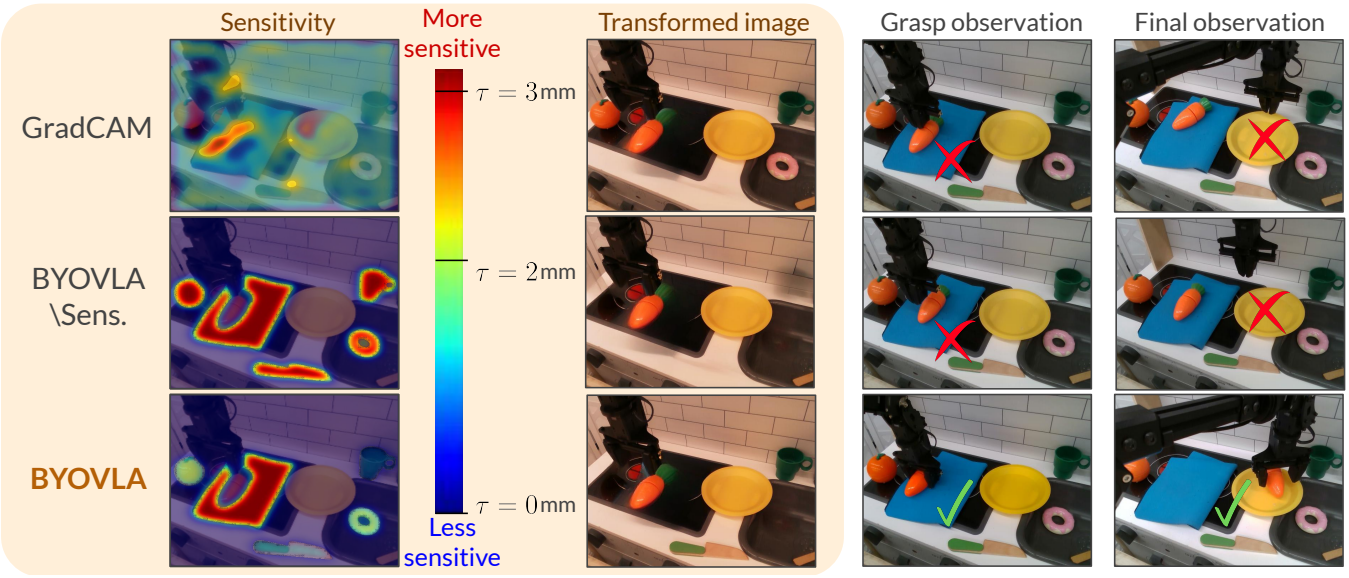

Fig. 3: Visualization of exemplar object distractors for the task of "put the carrot on yellow plate." First column: heatmaps showing the regions each method deems the VLA is sensitive to. Second column: inpainted image regions with sensitivity threshold τ. BYO*VLA* inpaints the the blue towel, orange, and donut, and then successfully grasps the carrot and puts it on the plate (last two columns), while BYO*VLA*\Sens. additionally inpaints the green knife and cup, but fails the task. GradCAM fails to capture the model sensitivity to most irrelevant objects and thus also fails.

architecture (layer 6 in Octo-Base), which is motivated by recent work in mechanistic interpretability suggesting that intermediate layers of transformer-based architectures contain salient features [43, 44, 45, 46]. To determine which image regions to manipulate, we compute the difference between maximal and minimal GradCAM scores and retain the pixel locations corresponding to the top quarter fraction. See Fig. 3 for an example of the regions deemed salient in an environment similar to those presented in Fig. 2.

We few-shot prompt GPT4-o with five in-context examples including images containing our kitchenette environment with distractions, along with a list of the task-irrelevant regions in the image. At run-time, we only query GPT4-o with the initial observation o_0 and ask for the task-irrelevant in terms of a list of objects and backgrounds. Unless stated otherwise, determination of task-irrelevant regions with GPT4-o is done once at initialization and visual sensitivity probing is performed at every timestep.

A. *Evaluation with Octo-Base*

Task and distractors. BYO*VLA* is first evaluated with Octo-Base on the task "place the carrot on yellow plate." The bottom images of Fig. 2 depict the environmental distractions present in the kitchenette environment. The left scene depicts the environment without distractions, and the middle scene showcases five task-irrelevant objects: orange, blue sponge, knife, green cup, and donut. The rightmost scene demonstrates the yellow tiling background distraction mentioned previously.

Implementation details. For the object distractor experiments, 15 trials are completed for each baseline. For visual sensitivity probing, we Gaussian blur object regions with a random kernel size between 15 and 30, and add Gaussian noise $\eta \sim \mathcal{N}(0, \sqrt{0.075})$ to the RGB channels for background regions. The threshold is set to $\tau = 2$mm for object and background regions for reasons described in Sec. III. We utilize the full extent of Octo's action chunking capability ($T_a = 3$), which we find most effective for achieving the baseline success rate reported in [3, Appendix]. In calculating Eq. (2), $K = 5$ rollouts are sampled.

Results. As Fig. 2 shows, task-irrelevant object distractions drop Octo-Base's success rate by 60%. Manipulating the image via inpainting following GradCAM fails to retain the nominal task success rate of 67% without distractions present, whereas BYO*VLA* and BYO*VLA*\Sens are able to retain or surpass the nominal success rate.

For background distractions, a similar trend in performance is observed. In this case, the GradCAM baseline also outperforms Octo-Base while BYO*VLA* and BYO*VLA*\Sens achieve the best performance, raising Octo's task success rate by ∼50%.

In general, the most common failure modes observed across all distractions and all baselines, BYO*VLA* included, are early grasping of the task object and missing the task object upon approach (see Fig. 3 for a visualization similar to the object distractions in Fig. 2). These failure modes are especially reflective of the effect of distractions since we only vary the initial position of the task object by 1-3cm from its central location between trials. Since this variation is smaller than the gripper's width when open, it highlights the deleterious effect distractions have on the policy.

The improvement of BYO*VLA* over BYO*VLA*\Sens. in the object distractor experiments is likely attributable to the presence of distractors in the training data, e.g., the BridgeV2 dataset, which forms a significant portion of Octo's training data. By removing all such distractions from the

input image, the distribution shift induced by inpainting is likely responsible for the policy's failure. On the other hand, we find that GradCAM is not attending to all regions relevant for Octo's output; otherwise, the success rate would have approached the nominal. The rollout visualization in Fig. 3 depicts a failed trajectory due to an early grasp of the object when inpainting what GradCAM says Octo is sensitive to, whereas running BYO*VLA* results in a successful trajectory.

B. Evaluation with OpenVLA

Task and distractors. We next study BYO*VLA* with OpenVLA on the task "put the eggplant in the pot," where we seek to answer two questions: (1) to what extent is BYO*VLA* model-agnostic, and (2) can BYO*VLA* offer any benefit to policies that build on vision-language models pretrained on large-scale internet data? We again utilize distractor objects from the BridgeV2 dataset, which accounts for roughly a sixth of total data that OpenVLA was trained on [5]. The rightmost image in Fig. 4 depicts the task-irrelevant objects: blue sponge, knife, green cup, and donut. The yellow tiling again was chosen to contrast the original white tiling as a background distraction.

Implementation details. Object (background) regions are again Gaussian blurred (Gaussian noised) for visual sensitivity probing with a threshold of $\tau = 2$mm for objects and $\tau = 2$mm for background regions (identical to the Octo experiments). Unlike Octo, OpenVLA does not action-chunk its commands. Moreover, unlike Octo which uses a diffusion policy and outputs stochastic action chunks, OpenVLA deterministically outputs the autoregressed action. As in the Octo experiments, a value of $K = 5$ is used for evaluating Eq. (2).

Results. As Fig. 4 shows, while OpenVLA nominally achieves a near perfect task success rate, the presence of distractions dropped its performance by 20%. BYO*VLA* is able to nearly achieve the nominal task success rate of in the presence of distractions, improving the baseline performance by approximately 15%. On the other hand, we observe that BYO*VLA*\Sens. achieves a worse task success rate, which is hypothesized to result from the distribution shift incurred from removing all task-irrelevant objects. A common failure mode observed for OpenVLA across all baselines, BYO*VLA* included, was the tendency to not command any changes in position after successfully grasping the task object, e.g., not lifting the eggplant off the stove. Likewise, many rollouts executed large downward vertical commands into the toy kitchen environment, which were prematurely ended to prevent hardware damage.

While BYO*VLA* does not retain the nominal success rate of OpenVLA with distractions, the environment depicted in Fig. 4 is significantly more challenging than the environment for Octo's experiments, as it contains both object and background distractions.

V. Conclusion and Discussions

We present BYO*VLA*: a run-time intervention scheme that dynamically determines task-irrelevant regions that an arbitrary VLA is sensitive to and minimally alters the image with automated image editing tools to improve policy performance in the presence of object and background distractions. BYO*VLA* is applicable off the shelf and does not require access to the VLA's weights. Experiments show that BYO*VLA* allows VLAs to nearly retain their nominal performance in the presence of task-irrelevant distractors, which otherwise drop the task success rate by up to 60%.

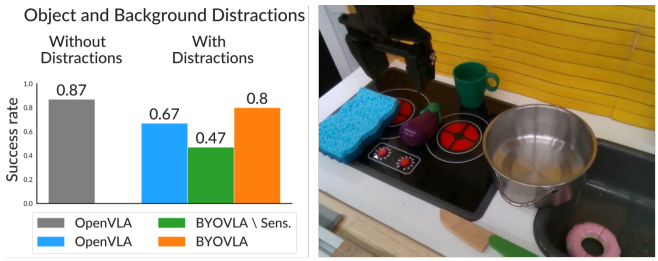

Fig. 4: First column: task success rates for BYO*VLA* with OpenVLA on language instruction "put the eggplant in the pot." Second column: kitchenette environment from BridgeV2 dataset with distractions.

Limitations and future work: The success of BYO*VLA* is reliant upon orchestrating different foundation models into a common pipeline, which presents challenges with integration. One limitation of our approach is the distinction between object and background distractions; while VLMs like GPT4-o can in principle discern between the two, our experiments primarily focus on cases where objects and backgrounds are easily separated to maximize the performance of GPT4-o in determining task-irrelevant regions. We expect that as VLMs become more capable, this aspect of BYO*VLA* improves.

Moreover, the regions proposed by the VLM are not guaranteed to be found with a separately trained segmentation model, and the choice of threshold τ is a hyperparameter of our method that requires a few real-world deployments to fine-tune for best results. Future work will consider how to better choose a threshold for a given environment, e.g., using conformal prediction [47, 48] to bound the false positive rate of detecting sensitive regions. In addition, we plan to explore more sophisticated and efficient inpainting schemes for background regions that seek to replace the background at deployment time with backgrounds from the VLA's training data. Finally, we only consider static environments in this work and plan to apply BYO*VLA* in *dynamic* environments, e.g., with a mobile robot or human behavior in the scene, where task-relevancy may change during policy execution. Therein, we expect that running the entire pipeline of BYO*VLA* at every time-step will offer advantages not capitalized on here.

Overall, we believe that *run-time interventions* — as a form of test-time compute — represent an underexplored avenue for significantly improving the base capabilities of VLAs without additional training, and we hope that the work presented here spurs further research in this area.

REFERENCES

[1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "RT-1: Robotics transformer for real-world control at scale," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[4] 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," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[5] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[6] 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 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[7] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[8] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Proceedings of Conference on robot learning (CoRL)*, 2023.

[9] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[11] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[12] Z. Yuan, T. Wei, S. Cheng, G. Zhang, Y. Chen, and H. Xu, "Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning," *arXiv preprint arXiv:2407.15815*, 2024.

[13] T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter *et al.*, "Scaling robot learning with semantically imagined experience," in *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[14] L. Y. Chen, C. Xu, K. Dharmarajan, Z. Irshad, R. Cheng, K. Keutzer, M. Tomizuka, Q. Vuong, and K. Goldberg, "Rovi-aug: Robot and viewpoint augmentation for cross-embodiment robot learning," *arXiv preprint arXiv:2409.03403*, 2024.

[15] Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar, "CACTI: A framework for scalable multi-task multi-scene visual imitation learning," *arXiv preprint arXiv:2212.05711*, 2022.

[16] B. Grooten, T. Tomilin, G. Vasan, M. E. Taylor, A. R. Mahmood, M. Fang, M. Pechenizkiy, and D. C. Mocanu, "Madi: Learning to mask distractions for generalization in visual deep reinforcement learning," in *Proceedings of International Foundation for Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

[17] M. Riedmiller, T. Hertweck, and R. Hafner, "Less is more–the dispatcher/executor principle for multi-task reinforcement learning," *arXiv preprint arXiv:2312.09120*, 2023.

[18] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitkovich, F. Xia *et al.*, "Open-world object manipulation using pre-trained vision-language models," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[19] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[20] J. Yang, W. Tan, C. Jin, K. Yao, B. Liu, J. Fu, R. Song, G. Wu, and L. Wang, "Transferring foundation models for generalizable robotic manipulation," *arXiv e-prints*, pp. arXiv–2306, 2023.

[21] Y. Miyashita, D. Gahtidis, C. La, J. Rabinowicz, and J. Leitner, "Roso: Improving robotic policy inference via synthetic observations," *arXiv preprint arXiv:2311.16680*, 2023.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[23] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[24] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of self-interpretable agents," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[25] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, 2022.

[26] V. Pacelli and A. Majumdar, "Learning task-driven control policies via information bottlenecks," in *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[27] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, "Generalization in reinforcement learning with selective noise injection and information bottleneck," *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[28] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[29] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[30] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017.

[31] 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.

[32] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[33] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[34] P.-J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (un) reliability of saliency methods," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, 2019.

[35] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[36] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[37] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[38] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[39] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan, Z. Zeng, H. Zhang, F. Li, J. Yang, H. Li, Q. Jiang, and L. Zhang, "Grounded sam: Assembling open-world models for diverse visual tasks," 2024.

[40] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," in *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[41] T. Yu, R. Feng, R. Feng, J. Liu, X. Jin, W. Zeng, and Z. Chen, "Inpaint

anything: Segment anything meets image inpainting," *arXiv preprint arXiv:2304.06790*, 2023.

[42] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Proceedings of Conference on Robot Learning (CoRL)*, 2023.

[43] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, and T. Henighan, "Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet," *Transformer Circuits Thread*, 2024. [Online]. Available: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html

[44] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu, "Scaling and evaluating sparse autoencoders," *arXiv preprint arXiv:2406.04093*, 2024.

[45] N. Elhage, T. Hume, C. Olsson, N. Nanda, T. Henighan, S. Johnston, S. ElShowk, N. Joseph, N. DasSarma, B. Mann, D. Hernandez, A. Askell, K. Ndousse, A. Jones, D. Drain, A. Chen, Y. Bai, D. Ganguli, L. Lovitt, Z. Hatfield-Dodds, J. Kernion, T. Conerly, S. Kravec, S. Fort, S. Kadavath, J. Jacobson, E. Tran-Johnson, J. Kaplan, J. Clark, T. Brown, S. McCandlish, D. Amodei, and C. Olah, "Softmax linear units," *Transformer Circuits Thread*, 2022, https://transformer-circuits.pub/2022/solu/index.html.

[46] H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *Foundations and Trends in Machine Learning*, 2023.

[48] G. Shafer and V. Vovk, "A tutorial on conformal prediction." *Journal of Machine Learning Research (JMLR)*, 2008.

APPENDIX

A. VLM Prompting

We provide an example of few-shot prompting GPT4-o to determine the task-irrelevant objects from the experiment "place the carrot on yellow plate" in Sec. IV. Each demonstration consists of a list of task-irrelevant objects, backgrounds, a natural language instruction, and an initial image. We utilized the same toy kitchen environment for all examples, but arbitrary environments from any offline dataset should suffice, e.g., BridgeV2 [42].

```
You are an assistant helping a robot determine what
    objects in the image are relevant for completing its
    task. You will be shown some text and images.

Example 1. Task: 'put strawberry in the bowl'

["hot dog", "broccoli"]
["white counter", "wall", "stovetop", "black sink"]
```

```
Example 2.
⋮

Example 5. Task: 'pick up the tomato'

["pizza", "olive oil", "lid"]
["wall", "counter", "stove", "sink"]
```

```
The robotic arm in the image is given the following task:
    'place the carrot on yellow plate.'
```

```
Provide a list of objects in the image that are not
    relevant for completing the task, called '
    not_relevant_objects'. Then provide a list of
    backgrounds in the image that are not relevant for
    completing the task, called 'not_relevant_backgrounds
    '. Give your answer in the form of two different
    lists with one or two words per object.
```

B. GradCAM Calculation

GradCAM is a model-specific calculation and we study its efficacy for sensitivity probing with Octo-Base as our VLA.

Given observation o_t, we perform one forward-pass through Octo-Base to obtain the multi-headed, cross attention weights $A_{i,j}^{(h,l)}$ between the task tokens i and image tokens j at head h: any extraneous tokens between 'wrist' or 'readout' tokens are removed (see [3] for further discussion regarding these token types). We then select the attention weights corresponding to the specific transformer layer l of interest and average across the task tokens dimension, yielding $A_j^{(h)}$.

For the gradient calculation, we perform one backward-pass through the VLA with the current observation o_t to obtain the multi-headed, cross attention gradient weights $\partial A_{i,j}^{(h,l)}$ between the task and image tokens, again removing extraneous tokens. We then average the gradients across the task token dimension at layer l and resize into a square array of dimension $\sqrt{d_{o_t}}$, the pixel dimensions of the current observation (256 in the case of Octo-Base). This yields the gradient calculation $\partial A_j^{(h)}$.

Finally, we compute the standard GradCAM calculation from [6] and average across the H attention heads h, yielding

$$L_{\text{GradCAM},j} = \frac{1}{H} \sum_h \left(\partial A_j^{(h)} \right) A_j^{(h)}. \tag{3}$$

In Octo-Base, there are 256 image tokens j, so the resultant GradCAM calculation in Eq. (3) is a 16x16 square matrix.

We apply a Gaussian blur to the GradCAM mapping to create a smoother image. An example of the GradCAM output and mask with thresholding a quarter of the greatest values can be seen in Fig. 5

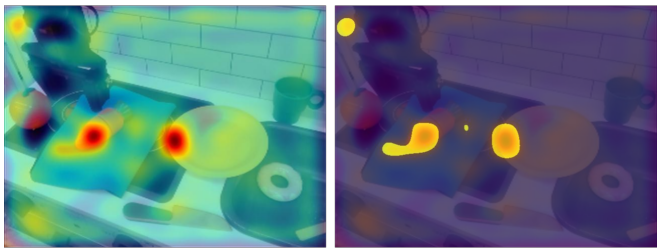

Fig. 5: Left: GradCAM output at layer 6 with Octo on language instruction "place the carrot on yellow plate." Right: mask used to select what objects were attended to by keeping the top quarter of GradCAM values.

C. Foundation Model Hyperparameters

1) Selection of Threshold τ: Selection of τ is described in Sec. III, which is based upon applying BYO*VLA* offline to environments from the BridgeV2 dateset. Fig. 6 showcases the third quartile, giving us an initial threshold of 5 [mm].

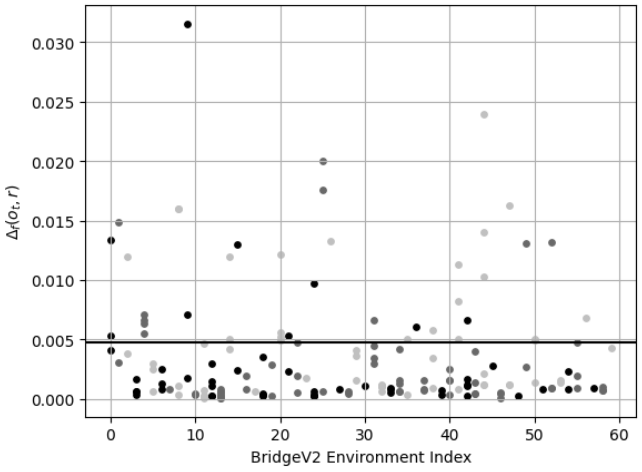

Fig. 6: Computation of Eq. (2) for BridgeV2 environments using the methodology prescribed in Sec. III. Each datapoint for a particular environment corresponds to a single perturbed task-irrelevant object. The horizontal line corresponds to the third quartile, arriving at a value of approximately $\tau = 0.005$.

2) Segmentation Model: We utilized Grounded-SAM2 with a confidence score of 0.4 for the box and text thresholds. See Fig. 7 for a demonstration of the output. For removing objects in image with [41], we used a mask dilation size of 10 which was empirically chosen to minimize image aberrations.

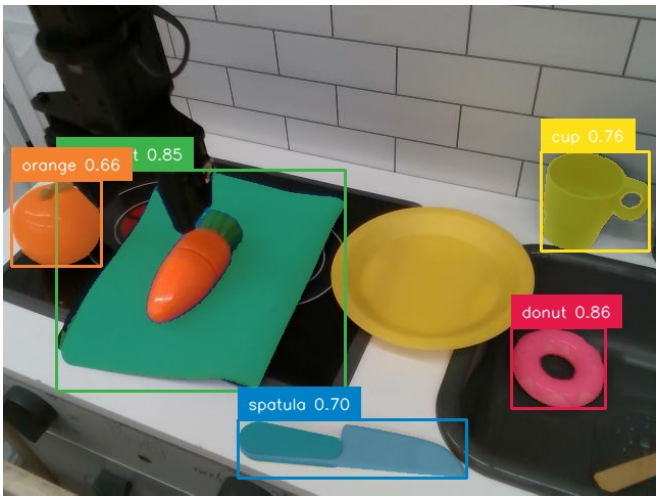

Fig. 7: Output from Grounded-SAM2 with Octo on language instruction "place the carrot on yellow plate." The output from the VLM was the list ['orange', 'blue mat', 'spatula', 'donut', 'cup'].