Visually Robust Vision-Language-Action Models Via Run-time Observation Interventions

Asher J. Hancock[1], Allen Z. Ren[1], and Anirudha Majumdar[1,2]

***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 in order 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 40%.**

I. Introduction

A longstanding goal in robotics research 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]–[4]. These models leverage their internet-scale training in order 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]–[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?*

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, and only requires a few seconds at initialization for diagnosis and observation editing before the rollout starts. Across multiple language-instructed manipulation tasks and varying distractor objects and backgrounds, BYO*VLA* improves task success rates by $20 - 40\%$ 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]–[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, unseen object appearances, or unseen 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]Princeton University, [2]Google DeepMind. Contact: ajhancock@princeton.edu

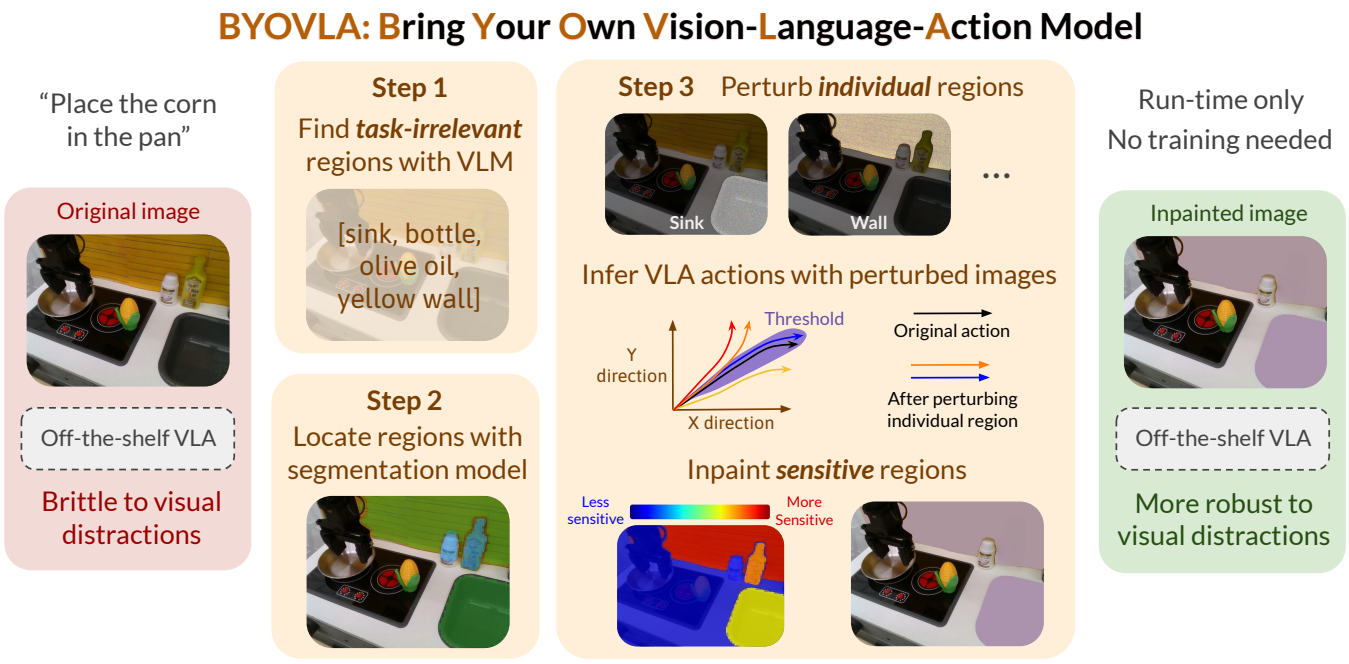

Fig. 1: We introduce BYO*VLA*: a simple and lightweight run-time intervention scheme for improving the performance of an arbitrary VLA model in the presence of task-irrelevant distractions. Our method identifies task-irrelevant regions in the visual observation and minimally modifies regions that the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

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]–[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]–[19]. However, simple masking can make the image observation look unrealistic and the model needs to be trained with such masked images. A more recent work [20] uses a VLM to determine the relevant objects in the scene based on the task instruction, but again it masks out *all task-irrelevant regions* and requires training with both the original and edited images. BYO*VLA* instead applies *selective* masking based on the model's sensitivity. Such minimal edits keep the edited observations relatively realistic and close to the training data (as Fig. 1 shows), thus not requiring additional training and also mitigating the potential artifacts generated by inpainting, leading to the improved performance of BYO*VLA* compared to the baseline that inpaints all task-irrelevant objects in Sec. IV. To our knowledge, [21] is the only other work that performs run-time *only* intervention of the camera observation for manipulation policy, but it does not deal with distractors but instead replaces the *task-relevant* but novel object with one seen during training.

Other training strategies for improving model robustness include creating bottlenecks in the attention mechanism [22, 23] of the policy architecture in order to train the policy to focus on objects selectively [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 off-the-shelf VLAs, 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,

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)
    $(\tilde{a}_t, \ldots, \tilde{a}_{t+T_a}) = f(\tilde{o}_t, l)$
    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 $\rho_f(o_t)$

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 Section 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* improve the task success rate of the base model in the presence of distractions.

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 the majority of 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 processes the raw observation o_t in order to produce a new observation $\rho_f(o_t)$ that is then sent as input to the VLA to produce an action chunk (sequence) $\bar{a}_t$:

$$\bar{a}_t = (a_t, \ldots, a_{t+T_a}) = f(\rho_f(o_t), l), \tag{1}$$

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 o_0 are task-irrelevant. We utilize the state-of-the-art GPT4-o model from OpenAI and prompt the model with few-shot exemplars corresponding to images, language instructions, and irrelevant regions. The output from GPT4-o is a string of region proposals in the initial observation deemed task-irrelevant, which is provided to a grounded segmentation model [37]–[39] to localize and partition the regions at the pixel-level at every step of the roll-out. Fig. 1 depicts the outputs at each stage of the process. In our work, we utilized Grounded-SAM2 [37, 38, 40]–[43] for segmentation. We only consider static environments, so 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, 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 $(a_t, \ldots, a_{t+T_a}) = f(o_t, l)$ 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 displacement and pose of the end-effector 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 observations $\{o_{t,k}\}_{k=1}^{K}$ and compute an average weighted L_2-norm of the difference in actions $\Delta a_{t+t'} \coloneqq a_{t+t'} - \tilde{a}_{t+t'}$ (where $t' \in \{0, \ldots, T_a\}$):

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

where $w \in \mathbb{R}^7$ is a weighting vector.

While numerous methods exist to perturb an image, we consider Gaussian blurring (smoothening) for object distractions and Gaussian noising for background distractions. Our choice of perturbation reflects offline evaluations of Octo in environments from the BridgeV2 dataset [44], which includes the physical kitchenette we consider in our experiments. Empirically, we also find that this simple choice of perturbation distribution is sufficient to capture model sensitivity.

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. In order to determine τ for object distractions, we utilize

the first observation from 50 trajectories in BridgeV2 and apply Gaussian blurring to 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 first quartile, we arrive at a value of approximately 0.004m. Since different environments in BridgeV2 are of different physical scales, we adjust the threshold by rolling out a few trials in our kitchenette, ultimately arriving at a threshold value for object distractors of $\tau = 0.002\text{m}$.

For background distractions we repeat the same procedure and also arrive at the threshold of $\tau = 0.001$. We use Gaussian noise perturbations across the RGB channels of the observation, instead of Gaussian blurring, as we find blurring too weak to elicit a substantial deviation in trajectories for these regions. The values for τ described above are 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 is 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 utilize Inpaint Anything [45]. 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 region with, recalculate $Eq.$ (2) for the inpainted image and inpainted-plus-noised image, and repeat until we are below our threshold. We found this simple recipe to work well in practice.

IV. Experiments

We evaluate BYO*VLA* with two state-of-the-art open-source VLA models: Octo-Base [3] (a 93M parameter transformer-based diffusion policy) and OpenVLA [5] (a 7B transformer-based 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 [44] 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, 5-7 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 were 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 [44]. Camera angles and object/background distractions are held constant during all trials. A threshold of 0.7 for the gripper state was set to determine when to open or close the end-effector. The Widow X was positioned above the task object to ease policy execution. In accordance with [3] and [5], the task object's initial position was varied 1-3cm from its central position between trials. Unless otherwise stated, 15 trials for all baselines are completed.

Inference 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 took less than three seconds to complete and costed less than one cent with five few-shot exemplars; since we are assuming static environments, this query is executed once at the beginning of the episode. Octo-Base can perform 13 iterations per second and OpenVLA can perform 6 iterations per second on a NVIDIA GeForce RTX 4090 GPU [3, 5]. On average, localizing task-irrelevant objects with the segmentation model, applying the visual sensitivity probe with Gaussian blur perturbations over 5 objects with $K = 5$ samples, and inpainting the image required roughly 2 seconds to complete. All time measurements were averaged over 15 trials.

Baselines. We evaluate BYOVLA against two baselines: (1) the original VLA policy; (2) BYO*VLA* without the visual sensitivity probe, 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 compute GradCAM for the cross-attention mechanism for the image tokens attended to by the task tokens, averaged across the attention heads halfway through the overall transformer architecture (layer 6 for Octo-Base). The choice of layer is motivated by recent work in mechanistic interpretability suggesting that intermediate layers of transformer-based architectures contain salient features [46]–[48]. Likewise, prior work that utilizes GradCAM in transformer-based architectures select an intermediate layer for analysis [49]. To determine which regions in the image 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.

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

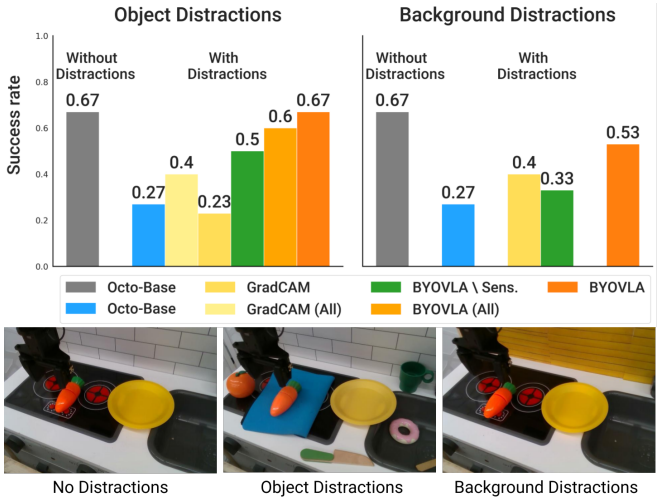

Fig. 2: First row: task success rates for BYO*VLA* with Octo on language instruction "place the carrot on yellow plate." Second row: kitchenette environment from BridgeV2 dataset with and without object and background distractions.

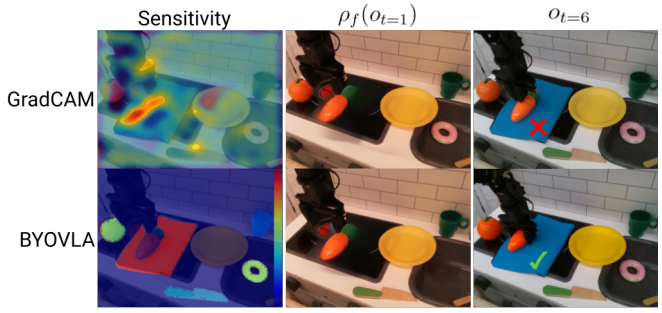

Fig. 3: First row: GradCAM output from Octo-Base model evaluated at the cross-attention mechanism between language tokens and image tokens at layer 6, along with its transformed image $\rho_f(o_{t=1})$ and resultant trajectory at $t = 6$. According to GradCAM, the VLA is only attending to task relevant regions; yet, the policy's brittleness to distractors implies the model is attending to these. Second row: sensitivity heat map from BYO*VLA*, along with its transformed image $\rho_f(o_{t=1})$ and resultant trajectory at $t = 6$. BYO*VLA* yielded a successful grasp and task completion.

distractions present in the kitchenette environment in our experiments. The leftmost scene showcases five task-irrelevant objects: orange, blue towel, knife, green cup, and donut, and the rightmost scene demonstrates the yellow tiling back distraction mentioned previously.

Implementation details. In principle, the distinction between an object and background distraction can be determined by GPT4-o. In our experiments with Octo, we considered object and background distractions separately, and hence did not prompt GPT4-o to distinguish between them. While rare, if GPT4-o erroneously listed either the task object (carrot) or goal state (yellow plate) as task-irrelevant, the trial was discounted and rerun.

For the object distractor experiments, 30 trials were completed for each baseline. For visual sensitivity probing, we Gaussian blurred object regions with a kernel size of 25, and added Gaussian noise $\eta \sim \mathcal{N}(0, \sqrt{0.075})$ to the RGB channels for background regions. The threshold was set to $\tau = 2$mm for objects and $\tau = 1$mm for background regions for reasons described in Sec. III. We transformed the input image with a warm filter in order to better match our physical operating conditions to Octo's training environments. We utilized the full extent of Octo's action chunking capability ($T_a = 4$), which we found most effective for achieving the baseline success rate reported in [3, Appendix]. In calculating Eq. (2), $K = 5$ roll-outs were sampled.

Results. Results are shown in Fig. 2. When task-irrelevant object distractors are present, Octo-Base's success rate drops by 40%. Naïvely manipulating the image via inpainting according to what GradCAM or GPT4-o suggest fails to reattain the nominal performance without distractions present. On the other hand, BYO*VLA* is able to achieve the nominal task success rate. A similar trend is observed when background distractions are employed. In this case, the GradCAM baseline slightly outperforms Octo-Base and BYO*VLA* achieves the best performance, raising Octo's task success rate by roughly 25%.

In general, the most common failure modes observed were early grasping of the task object (see Fig. 3) and missing the task object upon approach. The latter failure mode is especially reflective of the effect of distractions since we only varying the initial position of the task object by 1-3cm from its central location. Since this variation is smaller than the robotic gripper's width when open, it highlights the deleterious effect distractions have on the policy.

The improvement of BYO*VLA* over BYO*VLA* without the visual sensitivity probe is likely attributable to the presence of distractors in the training data. For instance, the BridgeV2 dataset — part of Octo's training data — often has object distractions in each environment. 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, in Fig. 3 we find that GradCAM is not attending to all regions relevant for Octo's output. The trial depicted resulted in a failed trajectory due to an early grasp of the object when inpainting what GradCAM said Octo was sensitive to, whereas use of BYO*VLA* resulted 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 which leverage vision-language models pretrained on large-scale internet data? OpenVLA exhibits significantly greater visual generalization capabilities than Octo [5], which will further elucidate any benefits offered by BYOVLA as models scale in size and complexity. We again utilized distractor objects from the BridgeV2 dataset, which

accounts for roughly a sixth of total data that OpenVLA was trained on. The rightmost image in Fig. 4 depicts the task-irrelevant objects: three silver lids, olive-oil, black pepper, grapes, and a pink plate. Brown bricks were chosen to contrast the original white tiling as a background distraction.

Implementation details. Object and background regions were again Gaussian blurred with kernel size of 25 for visual sensitivity probing with a threshold of $\tau = 2$mm for objects and $\tau = 1$mm for background regions (identical to the Octo experiments). Unlike Octo, OpenVLA does not action-chunk its commands, so a time-horizon of $T_a = 1$ was utilized. Moreover, since OpenVLA does not have a diffusion policy embedded in its architecture, only one roll-out ($K = 1$) was utilized for Eq. (2), since sampling with identical inputs yielded identical outputs.

Results. Fig. 4 summarizes the results. While OpenVLA nominally achieved a perfect task success rate, the presence of distractions dropped its performance by 40%. BYO*VLA* and its variant without sensitivity analysis both improved the baseline performance by 20-25% in the presence of distractions.

The most common failure mode observed for OpenVLA 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. Unlike Octo's behavior, we found OpenVLA to be more robust to initial position variations, further highlighting the non-intuitive effect of distractions on policy performance.

While BYO*VLA* did not retain the nominal success rate of OpenVLA with distractions, the environment depicted in Fig. 4 was significantly more cluttered than the environment for Octo's experiments. We found that even with few-shot prompting, GPT4-o often failed to locate all task-irrelevant objects, namely the lids located on the stovetop. Moreover, we found that due to the cluttered testing environment, GPT4-o struggled to accurately determine which regions ought to be considered objects and backgrounds, vice-versa. For this reason, we only few-shot prompted GPT4-o with object distraction exemplars and only inpainted sensitive object regions; this likely contributed to why BYO*VLA* did not reattain OpenVLA's nominal performance. Nonetheless, BYO*VLA* still conferred a significant benefit, and we anticipate further improvements in BYO*VLA* performance as the visual-reasoning capabilities of VLMs improve.

V. Discussion and Conclusions

We presented 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 distractions. BYO*VLA* is applicable off the shelf and does not require access to the VLA's weights and incurs minimal computational overhead. Experiments show that BYO*VLA* allows VLAs to nearly reattain their nominal performance in the presence of task-irrelevant distractor objects and background colors, which otherwise drop the task success rate by up to 40%.

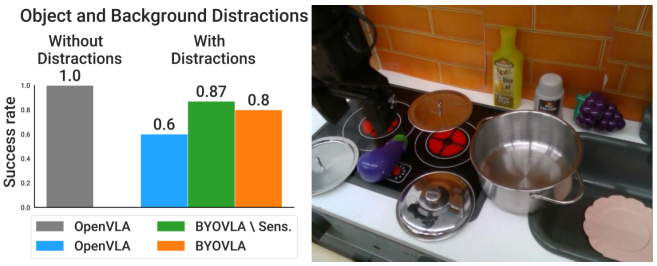

Fig. 4: First column: task success rates for BYO*VLA* with OpenVLA on language instruction "put the eggplant in the pot." Second column: kitchenette environment from BridgeV2 dataset with and without object and background distractions.

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 focused on cases where objects and backgrounds were 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* will improve.

Moreover, the regions proposed by the VLM have no guarantee of being found by a separately trained segmentation model. Our experiments focused on common household objects likely present in the training data of both the VLM and segmentation model, so this issue did not affect our results. However, in more complex environments, the probability that a segmentation model locates all of the VLM's region proposals may decrease. This issue may be alleviated with the future development of VLMs that are directly capable of segmentation.

The choice of threshold τ is a hyperparameter of our method that required 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 [50, 51] to bound the false positive rate of detecting sensitive regions. In addition, we will explore more sophisticated 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 considered static environments in this work, and plan to apply BYO*VLA* in *dynamic* environments, where task-relevancy may change during policy execution.

Overall, we believe that *run-time interventions* represent an underexplored avenue for significantly improving the base capabilities of VLAs, 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," *arXiv preprint arXiv:2212.06817*, 2022.

[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," *arXiv preprint arXiv:2307.15818*, 2023.

[3] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 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," *arXiv preprint arXiv:2310.08864*, 2023.

[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 the IEEE international conference on computer vision*, 2017, pp. 618–626.

[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 *Conference on robot learning*. PMLR, 2023, pp. 287–318.

[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," *arXiv preprint arXiv:2403.12945*, 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 *2017 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," *Advances in neural information processing systems*, 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," *arXiv preprint arXiv:2302.11550*, 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," 2024. [Online]. Available: https://arxiv.org/abs/2409.03403

[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," *arXiv preprint arXiv:2312.15339*, 2023.

[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," *arXiv preprint arXiv:2303.00905*, 2023.

[19] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," *arXiv preprint arXiv:2310.14386*, 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, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[23] D. Bahdanau, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[24] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of self-interpretable agents," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 414–424.

[25] S. James and A. J. Davison, "Q-attention: Enabling efficient learning for vision-based robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1612–1619, 2022.

[26] V. Pacelli and A. Majumdar, "Learning task-driven control policies via information bottlenecks," *arXiv preprint arXiv:2002.01428*, 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," *Advances in neural information processing systems*, vol. 32, 2019.

[28] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," *arXiv preprint arXiv:2006.10742*, 2020.

[29] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive behavioral similarity embeddings for generalization in reinforcement learning," *arXiv preprint arXiv:2101.05265*, 2021.

[30] S. Lundberg, "A unified approach to interpreting model predictions," *arXiv preprint arXiv:1705.07874*, 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, pp. 1135–1144.

[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*, vol. 33, no. 01, 2019, pp. 3681–3688.

[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*, pp. 267–280, 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)*, June 2024, pp. 26 296–26 306.

[37] 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.

[38] 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," *arXiv:2304.02643*, 2023.

[39] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen *et al.*, "Simple open-vocabulary object detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 728–755.

[40] 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

[41] 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," *arXiv preprint arXiv:2303.05499*, 2023.

[42] T. Ren, Q. Jiang, S. Liu, Z. Zeng, W. Liu, H. Gao, H. Huang, Z. Ma, X. Jiang, Y. Chen, Y. Xiong, H. Zhang, F. Li, P. Tang, K. Yu, and L. Zhang, "Grounding dino 1.5: Advance the "edge" of open-set object detection," 2024.

[43] Q. Jiang, F. Li, Z. Zeng, T. Ren, S. Liu, and L. Zhang, "T-rex2: Towards generic object detection via text-visual prompt synergy," 2024.

[44] 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 *Conference on Robot Learning*. PMLR, 2023, pp. 1723–1736.

[45] 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.

[46] 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

[47] 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.

[48] 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.

[49] 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*, 2021, pp. 782–791.

[50] A. N. Angelopoulos and S. Bates, "A gentle introduction to conformal prediction and distribution-free uncertainty quantification," *arXiv preprint arXiv:2107.07511*, 2021.

[51] G. Shafer and V. Vovk, "A tutorial on conformal prediction." *Journal of Machine Learning Research*, vol. 9, no. 3, 2008.

APPENDIX I
FULL VLM PROMPT

[**AR:** add VLM prompt]