A APPENDIX

A.1 INSTRUCTCV TRAINING DETAILS

We train our multi-task vision model across 20 epochs for 10 hours on an array of 8 80GB NVIDIA A100 GPUs. Our training utilizes images of 256×256 resolution and a batch size of 128. Augmentation techniques applied include random horizontal flipping and crop augmentation. For the latter, images are first subjected to random resizing between 256 and 288 pixels before being cropped to a 256-pixel size. We set our learning rate at 10e-4 without incorporating a learning rate warm-up phase. Model initialization is performed using the EMA weights from the Stable Diffusion v1.5 checkpoint, and we adopt other training settings from the public Stable Diffusion repository. For the results presented in this paper, we operate at a 256-pixel resolution with 100 denoising steps. We employ an Euler ancestral sampler with a denoising variance schedule as proposed by [59]. On an NVIDIA A100 GPU, our model takes approximately 10 seconds to solve a given vision task.

A.2 EXAMPLES OF THE REPHRASED PROMPTS.

Figure 5-(a)/(b) qualitatively compares the robustness of InstructCV to changes in instruction wording when trained using the fixed prompt or the more diverse rephrased prompt dataset. The model trained using the fixed prompt data shows reasonable performance on segmentation tasks, even for prompts that slightly deviate from the standard instruction wording. However, as these deviations increase misclassfications become more common. Instead, the model trained on the rephrased prompt data appears more robust to such changes in task formulation. Notably, the model appears to show basic semantic understanding as it is in the last prompt example able to infer the correct intent despite the simultaneous occurrence of the potential object detection targets 'spider man' and 'face'.

A.3 POST-PROCESSING STEPS FOR OBJECT DETECTION TASKS

We employ image processing techniques to derive the bounding box coordinates from the output image. First, we apply median and bilateral filters to the image in order to mitigate noise and enhancing features. Following, we convert the image from RGB to HSV space to isolate the red region within the target object, which is then extracted from the original image. Next, we identify closed contours in the image by converting it to a grayscale map, performing threshold segmentation based on grayscale, and ultimately, binarizing the image. However, naively applying this approach could potentially result in the removal of some accurate bounding boxes due to the disruptions induced by complex image backgrounds. To circumvent this issue, we cross-reference the bounding boxes with the dataset annotations and retain any bounding box predicted by the model that exhibits an Intersection over Union (IOU) greater than 0.5. We exclude disturbances that are not rectangular or that contain numerous red dots within the contour. The coordinates of the remaining contours are subsequently added to the prediction list.

Table 4: **Rephrased prompts**. We automatically generate diverse rephrased prompts to generated a diverse rephrased prompts (RP) instruction dataset. Here we display some examples of rephrased task templates.

	Segmentation	Detection	Classification	Depth estimation
Prompts	Segment the %.	Detect the %	Show * if there exists a % in the figure.	Estimate the depth of this image.
	Please break down the % into individual parts.	Can you help me detect the %?	Display an * if a % is present in the figure.	Approximate the depth of this image.
	Can you provide me with a segment of the %?	Please employ bounding boxes for the purpose of % detection.	In case a % exists in the figure, display an *.	Make an estimation of how deep the this image is.
	Please divide the % into smaller parts.	Locate the %'s presence.	If a % is present in the figure, indicate it with *.	Provide a rough calculation of the image's depth.
	Please perform image segmentation to isolate the % in this image.	Please use bounding boxes to identify the presence of a %.	If there is a % in the figure, show it as *.	Give an approximate measurement of the image's depth.
	Help me segment the %.	Detect and identify the %'s location.	Show * if the figure contains a %.	Make an informed guess of the depth of the image.
	Would you be willing to segment the %?	Utilize bounding boxes in order to identify the presence of the %.	Show an * if there is a % within the figure.	Make an estimation of how deep the image goes.

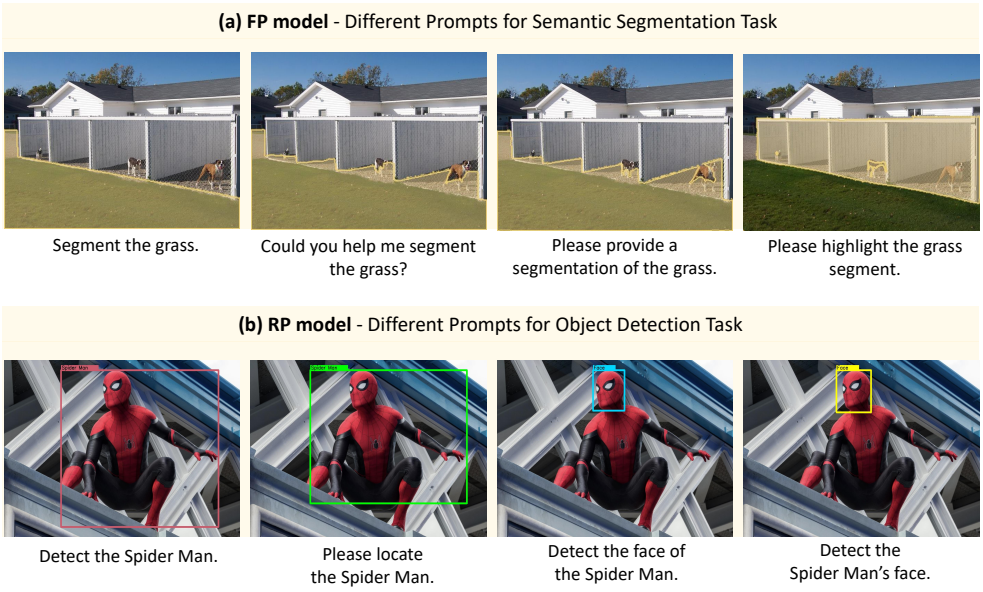

Figure 5: **Semantic segmentation performance for different prompt formulations** (a) **FP model**. Results for model trained on fixed prompt (FP) instruction dataset (b) **RP model**. Results for model trained on diverse rephrased prompt (RP) instruction dataset. Overall the RP model shows greater robustness to variations in wording.

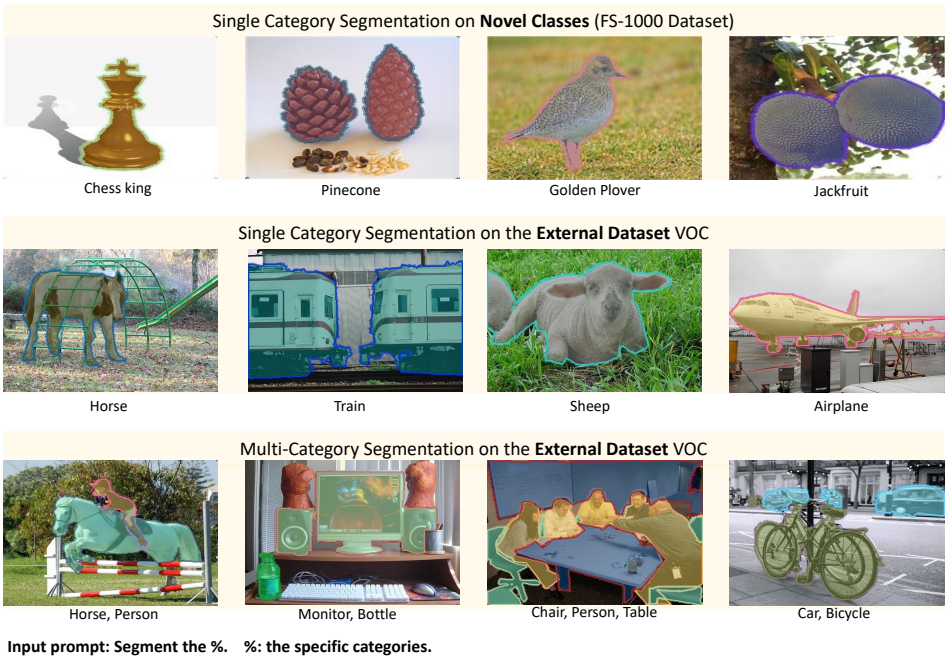

Figure 6: **Visualization of semantic segmentation examples.** Segmentation categories are provided below each image.

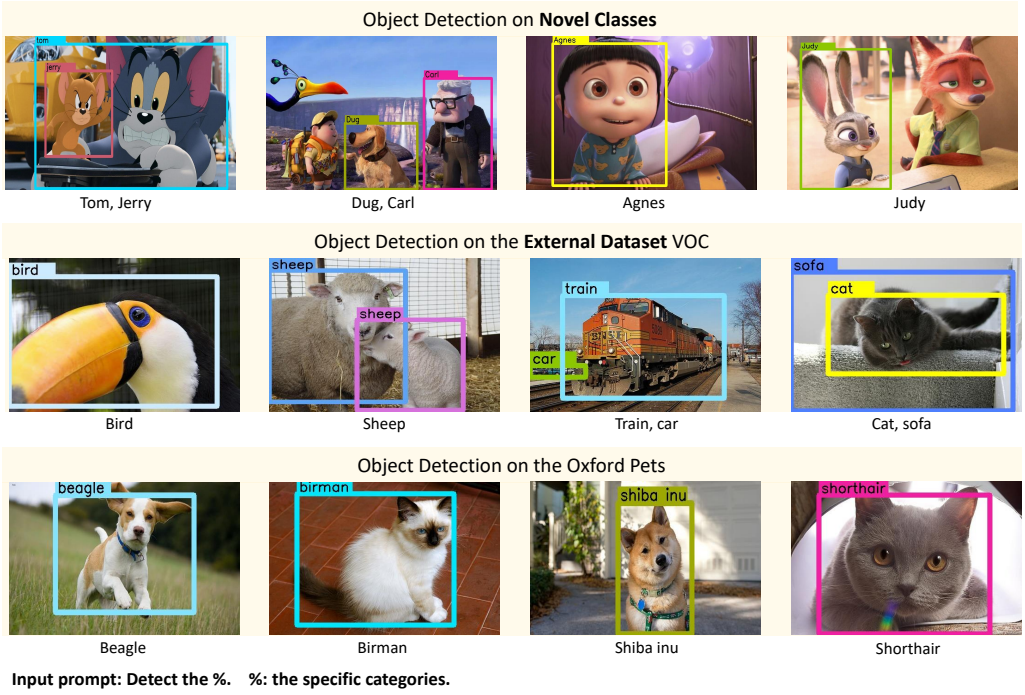

Figure 7: **Visualization of object detection examples.** Segmentation categories provided below the image.

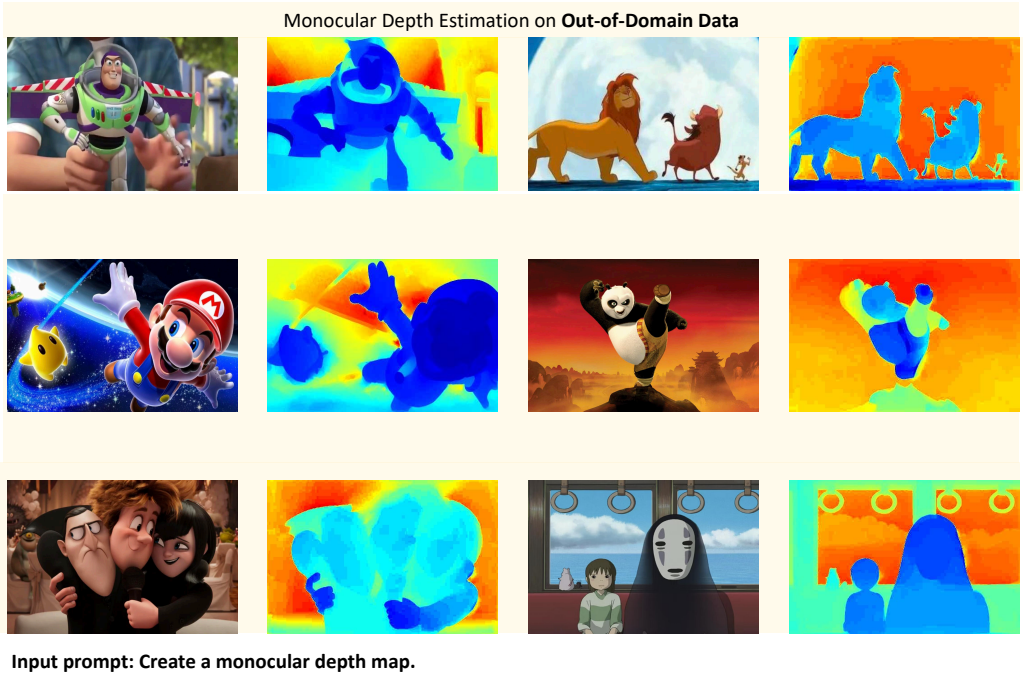

Figure 8: **Visualization of monocular depth estimation examples.** All depth maps presented here are produced for datasets not included during model training.