CObL: Toward Zero-Shot Ordinal Layering without User Prompting

Supplementary Material

S1. Details of synthetic generation pipeline

In the Blender 3D modeling step, we place random 3D assets from our library into the scene from front to back at regular depth intervals. Each object's horizontal location is sampled randomly from all candidate locations within the frame. Object locations that lead to total object occlusion are rejected and resampled. Once placements are determined, we extract all but one object and render shadows, depth and an object mask. Creating and rendering 750 scenes takes approximately 3 hours on a NVIDIA 3080 GPU. We do not render any images using the material maps associated with the objects in our library, which greatly reduces rendering time. But for visual comparison, Fig. S1 shows some examples of how those images would look, compared to the generated ones in Fig. S2.

In the texture generation step, we use the publicly available weights of pre-trained Controlnet-Depth conditioned with a rendered depth map and the prompt "a(n) {object_label}, in a well lit and empty room, {modifier_string}". Through trial and error and with help from ChatGPT we settled on the modifier string "hyperrealistic, photorealistic, 4k resolution, natural materials, realistic floor, realistic wall, intricate details, realistic color palette, HDR". We run 20 steps of DDIM guidance with a CFG scale of 7.5.

To generate diverse backgrounds, we once again query ChatGPT to end up with 20 possible background prompts. Example background prompts include:

- "A cozy indoor setting showcasing detailed floor textures, {modifier_string}"
- "An artistic interior scene with vibrant patterns on the floor, {modifier_string}"
- "A luxurious indoor environment with polished wooden floors, {modifier_string}"

We choose a background prompt at random, generate a scene conditioned on the input depth map, and assign it to the stack of object layers.

We then crop these outputs according to the masks to get candidate objects. Before compositing them together, we add all shadows to the background image x_1 by darkening pixels within the shadow regions. This update to x_1 is given by

$$x_1 \leftarrow x_1 \prod_{i=1}^{N} (1 - s_i), \tag{9}$$

where we approximate the effect of shadows as a compounding decrease in background albedo. We then composite the objects as explained in the main text, see Fig. S2 for examples of synthetic data.

S2. Architecture details

We augment the Stable Diffusion UNet architecture with an image conditioning adapter and lateral attention blocks. We run one instance of Stable Diffusion per object layer image, tied together with lateral attention. Each UNet receives the same input text condition, using the empty string ("").

After every spatial attention layer in the UNet, we add an input adapter corresponding to image and depth cues following [37]. The adapted cues are then passed equivalently to every concurrent UNet. We scale the adapter weighting by a learned parameter $\alpha_{in} \in [0, 1]$ such that the module is entirely skipped when $\alpha_{in} = 1$.

Lateral attention is applied as a combination of a Conv3D and inter-layer attention block, similar to the temporal attention implementation in [6]. In our case, we extend the size of the 3D convolution kernel to N as we find this helps propagate global information. Similar to the input conditioning block, each lateral attention block includes residual connection with a learned parameter $\alpha \in [0, 1]$, such that the block is skipped when $\alpha = 1$.

We find α has an additional benefit beyond scaling the effect of each additional block. During inference, we can temporarily set $\alpha, \alpha_{in} = 1$ to sample denoising estimates from base Stable Diffusion. This allows us to apply prior score matching without instantiating another model.

In total, CObL has 361 million trainable parameters, about 28% of the size of Stable Diffusion 2.1. The distribution of added model parameters across the different neural blocks are summarized in Sec. S2.

Network	Added Parameters
SD-UNet	183M
RGB Image Adapter	77.4M
Depth Map Adapter	77.0M

Table S1. Parameter counts for each model component.

S3. Non-differentiable guidance steps

We employ three non-differentiable steps during inference time at equally spaced times and we find they heuristically improve output quality. We apply these updates every 5 sampling steps of inference.

Permute At inference timestep t, we take all predictions $(\hat{x}^1_{0;t}, \ldots, \hat{x}^N_{0;t})$, exhaustively permute the ordering of these

predictions and compute the resulting composition of each permutation. We then choose the permutation that minimizes the compositional loss as described in Eq. (7). We find that this improves the quality of object layers of pairs of objects with very minor occlusions that may otherwise be sorted incorrectly. In practice objects with incorrect assignments will change permutations only the first time this operation is computed, but we repeat every 5 guidance steps as we find it rarely catches misclassifications in later stages of inference, with very little computational downside.

Erase As a non-differentiable update, we erase objects that are almost fully occluded. An object is considered fully occluded if less than 1% of the object is visible in the composited scene. At higher scales of classifier-free guidance, the latents are less constrained by the input scene and frequently hallucinate totally unrelated objects. If a hallucination happens to be fully occluded by previous object layers, it will not be penalized by either guidance loss. This update step removes these objects by replacing the object layer with an empty layer with zero alpha. In our experiments, we do not find that object hallucinations unrelated to the object layers occur outside of these cases.

Sort To maximize output quality, we ensure the object layers are generated in a similar ordinal structure to our training data. If our model generates layers without any objects, we note this layer as empty. We move the empty layers to the end of the layer stack so that for a scene with k non-empty layers, the last $N - k$ layers will be empty. We consider a layer as empty if its alpha is nonzero for less than 0.1% of the frame.

S4. Detailed Limitations

Beyond the high-level limitations discussed in Sec. 7, we find failure modes in CObL that can broadly be described as "merging" or "splitting" failure modes.

Merging occurs when the an object layer consists of multiple objects. We find this occurs more frequently with as the number of objects in an image increases. See the top row of Fig. S4.

Another point of failure is when an object is split across multiple layers, which generally occurs when the object has multiple visually distinct parts. In our opinion, these splits often create valid perceptual groupings. An example is shown in the bottom row of Fig. S4, where the mushroom object is split across two layers along its color discontinuity.

A possible reason these failures occur is due to our use of Stable Diffusion priors, which are meant for generation of large scale scenes and not objects in particular. If we instead replaced Stable Diffusion with a model that has a stronger single image prior this issue may be resolve. However, no such model currently exists that also contains the strong prior of Stable Diffusion,.

In addition, we use an off-the-shelf mask network for foreground segmentation during inference which leads to similar issues with model bias. We find the segmentation model can lead to incorrect mask assignments for complex objects with many parts, which leads to incorrect scene compositions. A mask network specifically trained for generating object layers may improve model performance on these more complex images.

S5. Initialization and Non-convexity

We find that the problem of determining correct object layers is extremely non-convex. As diffusion-based generated images, our model outputs are dependent on the sample of noise used as the initial latent z_T. While traditional diffusion approaches can leverage this randomness to increase output diversity, poor initialization for CObL latents can result in the failure modes discussed previously.

We find that this problem is more common in CObL than in other diffusion models because the initialization has increased total variance, as we run multiple diffusion models concurrently. However, we can leverage the exploratory nature of random initializations for improved object layer discovery.

On many real world scenes we qualitatively find that even for the incredibly non-convex problem of occlusion-ordered object layering we can arrive at a reasonable layer stack within a handful of initializations. This motivates the "best" and "average" comparisons in **??**. In practice, this is reflected by running CObL multiple times and choosing the most likely output.

The impact of initialization is visualized in Fig. S6. We show the effect of using the same initialization on scenes that have similar but distinct object layer representations. Even for scenes that have different numbers of total objects, we can see the same failure modes and successes repeated when using the same initialization. We also show that we achieve different estimated object representations for the scene depending on the initialization chosen. A similar result can be seen in Fig. S7.

Note that regardless of if a decomposition is "accurate" according to human perception, the object layers will generally composite back to the original scene.

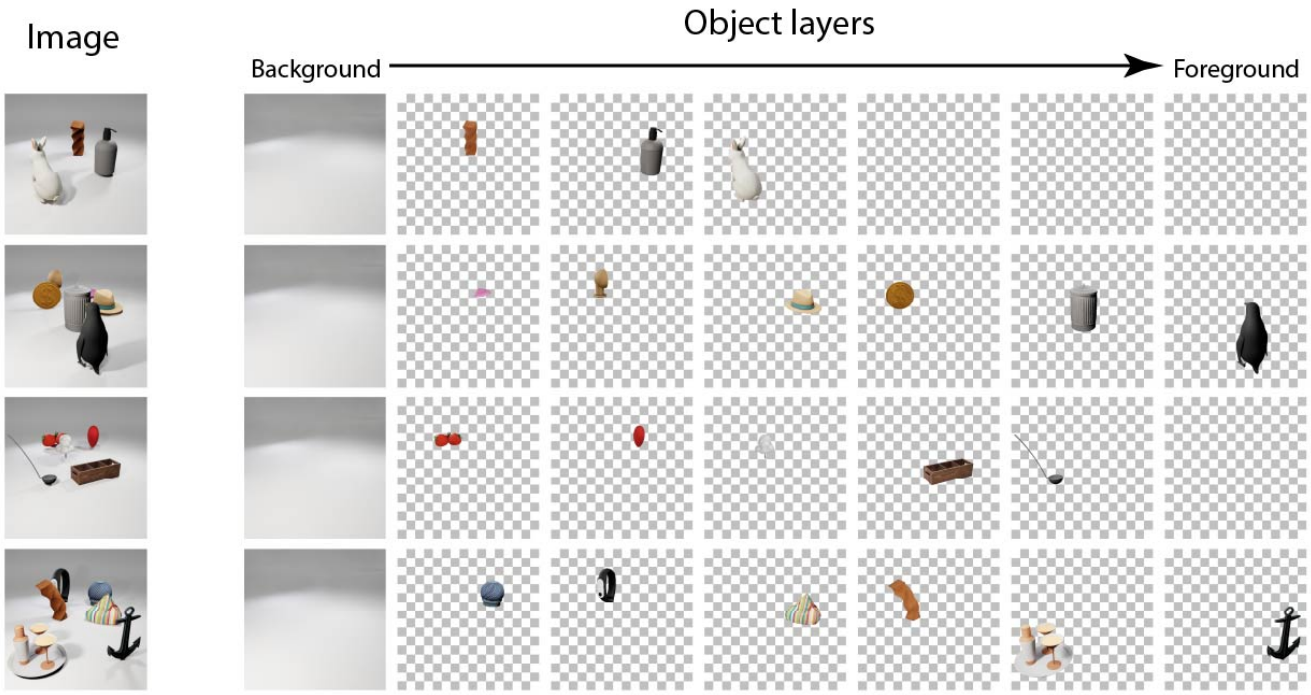

Figure S1. Renders of pre-diffusion textures. These renders are not used anywhere in the pipeline and are only shown for visualization purposes. In practice we extract the canonical object descriptors of each object for use in our pipeline, shown in Fig. 2.

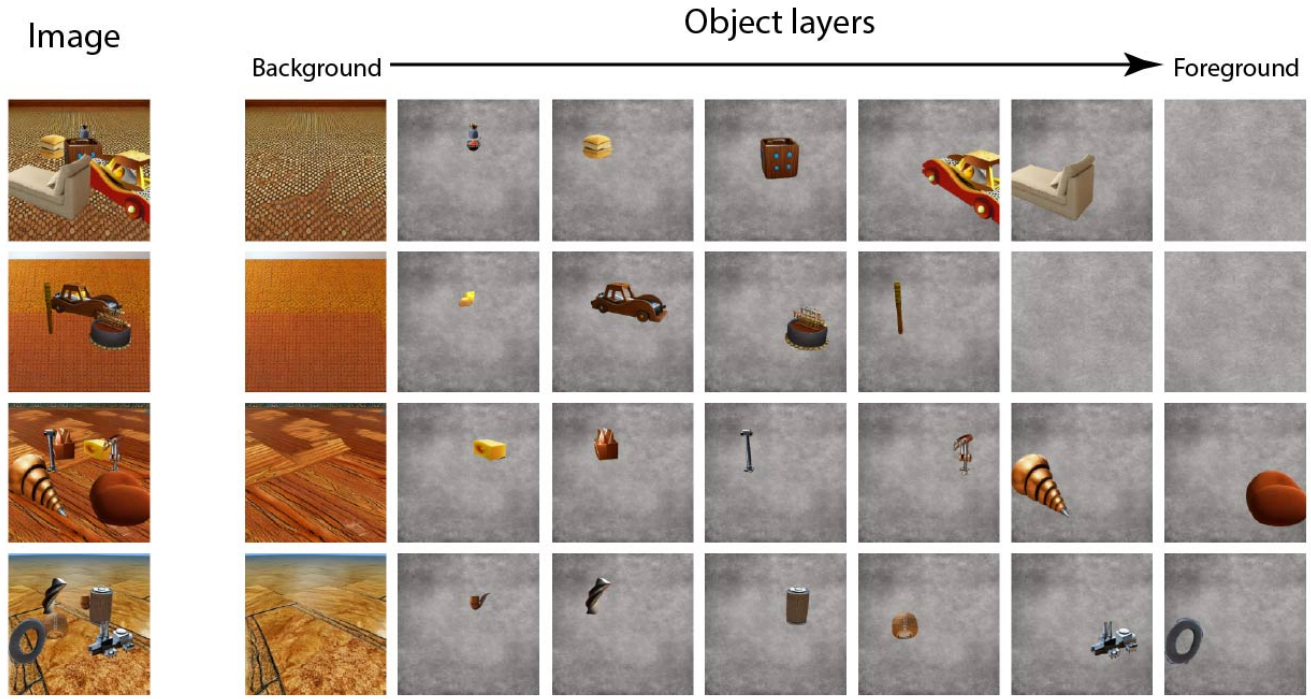

Figure S2. Random subset of our synthetic training dataset. Each object layer is placed on a gray background, and they can composite from back to front to form their corresponding image.

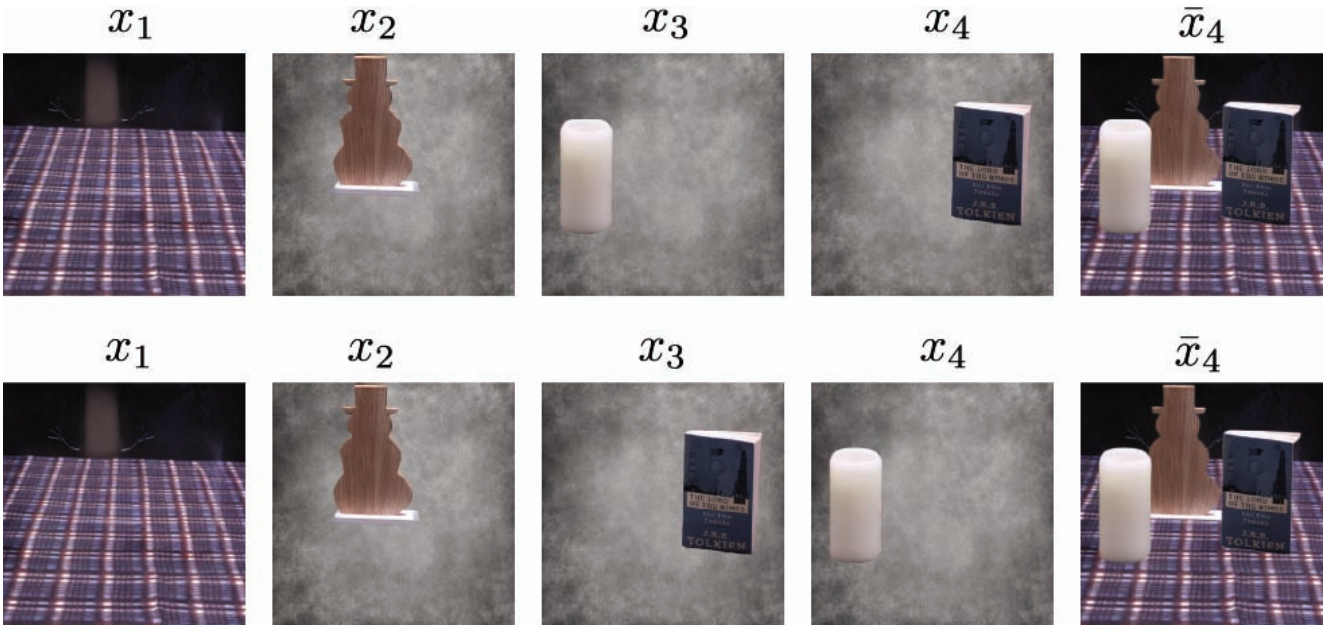

Figure S3. Two configurations of object layers that result in equivalent scene compositions. Both configurations are considered valid representations.

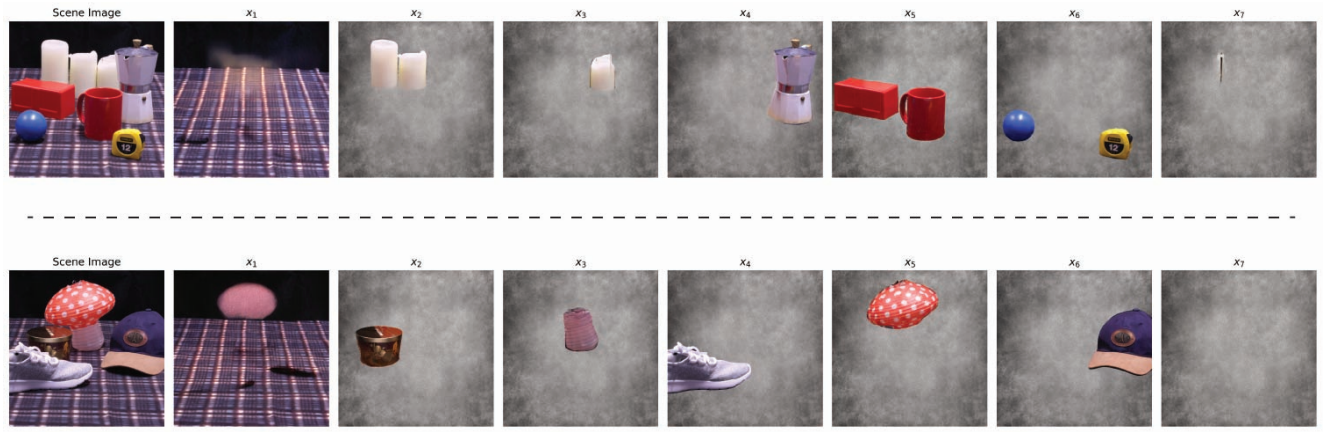

Figure S4. An illustration of the splitting and merging failure modes. (Top) A scene in which multiple disjoint objects are assigned the same layer. (Bottom) A scene in which an object (the mushroom lamp) is incorrectly characterized as two objects.

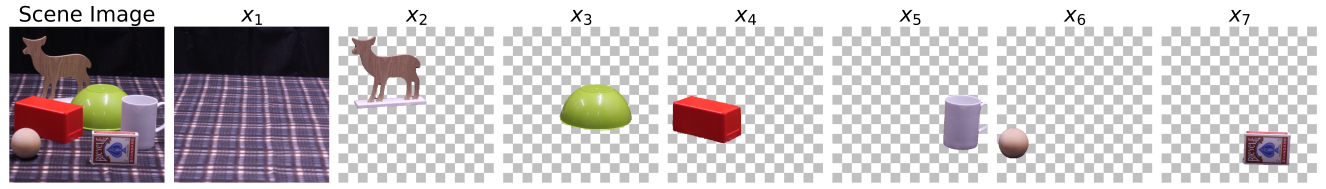

Figure S5. Ground-truth object layers for a 6 object scene in TABLETOP.

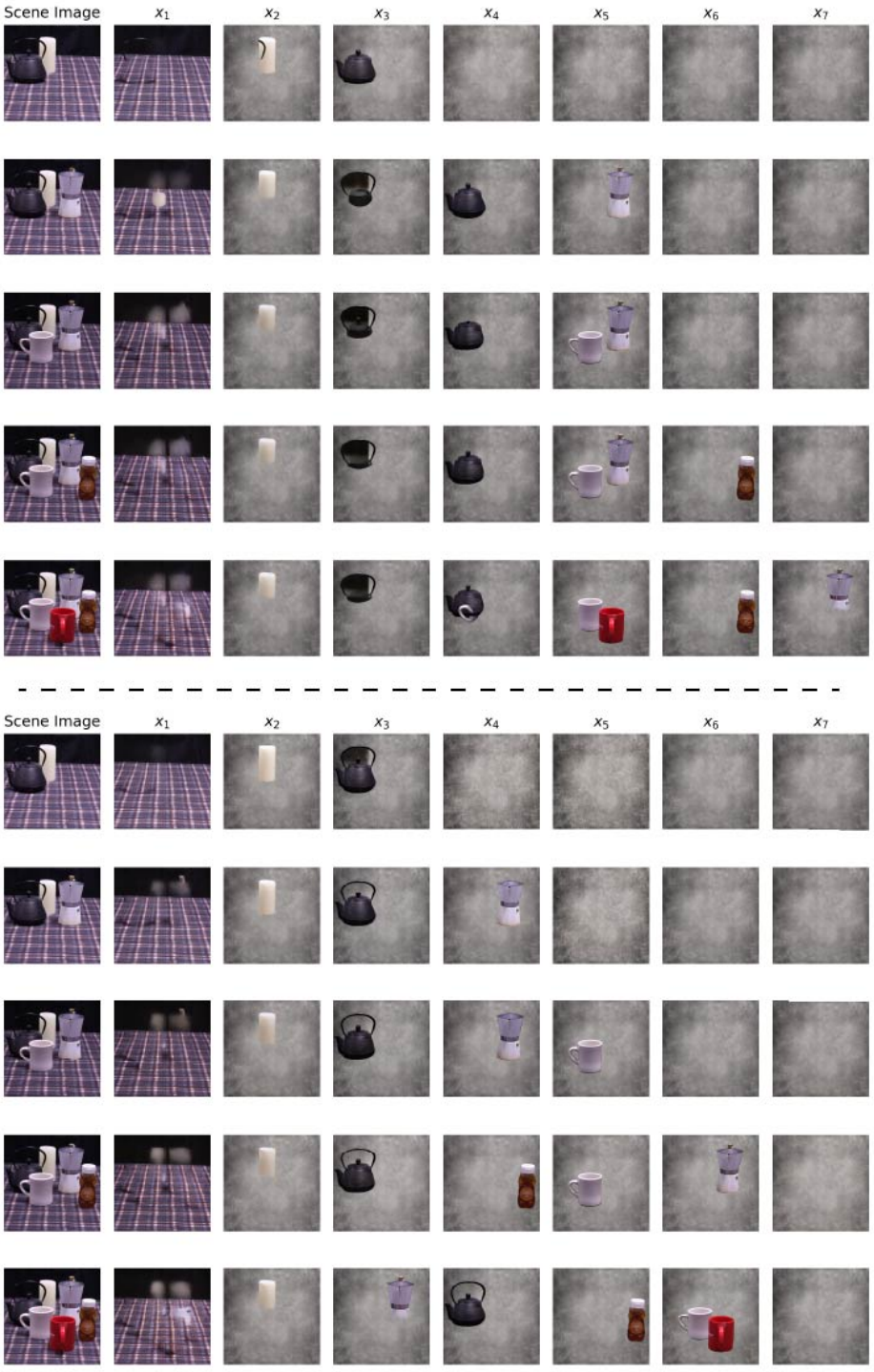

Figure S6. Learned object layers for two trials (seperated by the dashed line) of the same five scenes. In each trial, the latent initialization is kept consistent as the scenes change. Each scene varies from the previous by the addition of a single object.

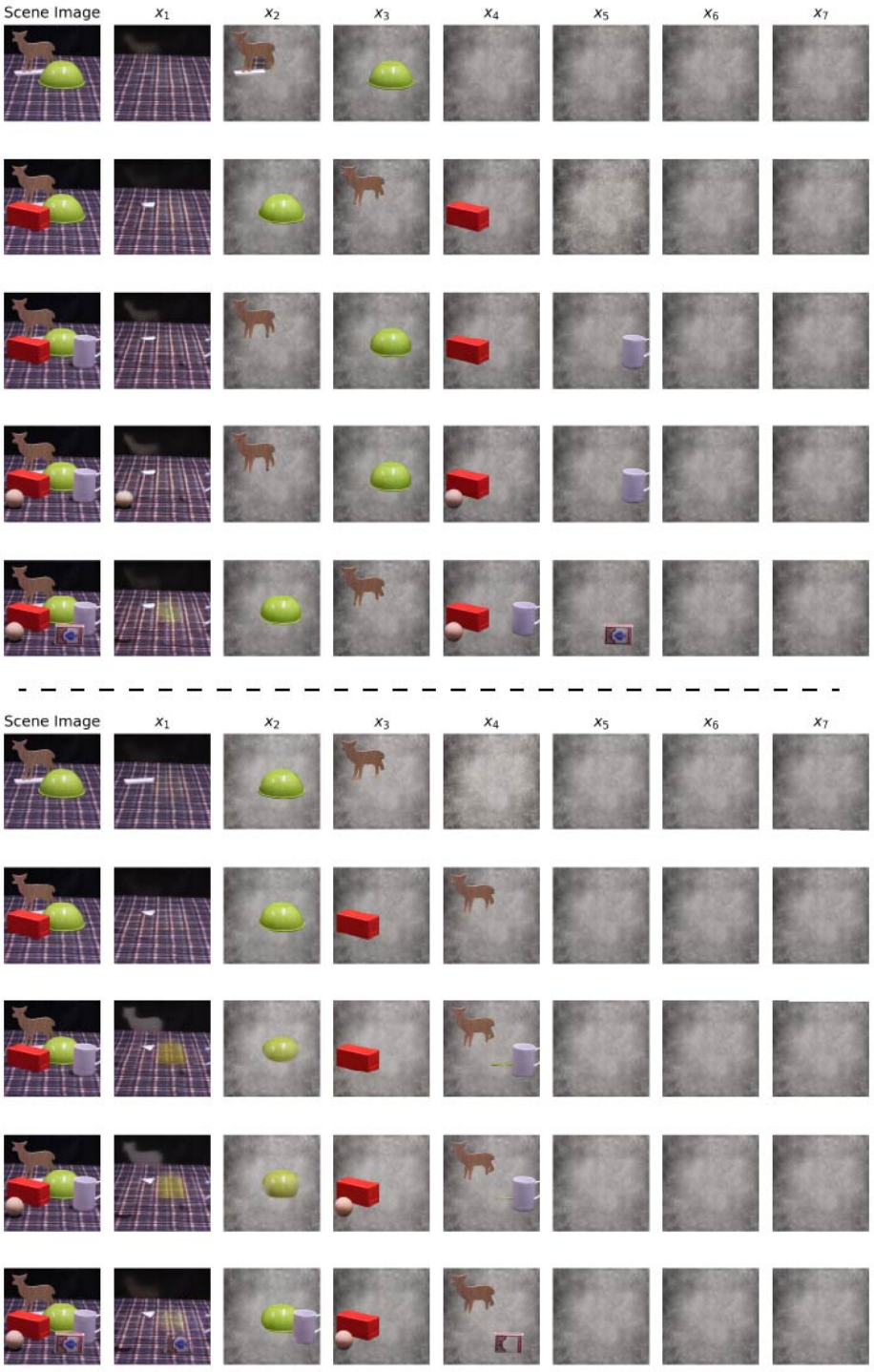

Figure S7. Learned object layers for two trials (seperated by the dashed line) of the same five scenes. In each trial, the latent initialization is kept consistent as the scenes change. Each scene varies from the previous by the addition of a single object.

Figure S8. The set of scenes in TABLETOP containing 6 objects.