Unsupervised Foreground Extraction via Deep Region Competition

Peiyu Yu[1] yupeiyu98@cs.ucla.edu
Sirui Xie[1] srxie@ucla.edu
Xiaojian Ma[1] xiaojian.ma@ucla.edu
Yixin Zhu[3] y@bigai.ai
Ying Nian Wu[2] ywu@stat.ucla.edu
Song-Chun Zhu[1,2,3] sczhu@stat.ucla.edu

[1]UCLA Department of Computer Science [2]UCLA Department of Statistics
[3]Beijing Institute for General Artificial Intelligence (BIGAI)

Abstract

We present Deep Region Competition (DRC), an algorithm designed to extract foreground objects from images in a fully unsupervised manner. Foreground extraction can be viewed as a special case of generic image segmentation that focuses on identifying and disentangling objects from the background. In this work, we rethink the foreground extraction by reconciling energy-based prior with generative image modeling in the form of Mixture of Experts (MoE), where we further introduce the learned pixel re-assignment as the essential inductive bias to capture the regularities of background regions. With this modeling, the foreground-background partition can be naturally found through Expectation-Maximization (EM). We show that the proposed method effectively exploits the interaction between the mixture components during the partitioning process, which closely connects to region competition [1], a seminal approach for generic image segmentation. Experiments demonstrate that DRC exhibits more competitive performances on complex real-world data and challenging multi-object scenes compared with prior methods. Moreover, we show empirically that DRC can potentially generalize to novel foreground objects even from categories unseen during training.[1]

1 Introduction

Foreground extraction, being a special case of generic image segmentation, aims for a binary partition of the given image with specific semantic meaning, *i.e.*, a foreground that typically contains identifiable objects and the possibly less structural remaining regions as the background. There is a rich literature on explicitly modeling and representing a given image as foreground and background (or more general visual regions), such that a generic inference algorithm can produce plausible segmentations ideally for any images without or with little supervision [1–8]. However, such methods essentially rely on low-level visual features (*e.g.*, edges, color, and texture), and some further require human intervention at initialization [4, 5], which largely limits their practical performance on modern datasets of complex natural images with rich semantic meanings [9, 10]. These datasets typically come with fine-grained semantic annotations, exploited by supervised methods that learn representation and inference algorithm as one monolithic network [11–16]. Despite the success of densely supervised learning, the unsupervised counterpart is still favored due to its resemblance to how humans perceive the world [17, 18].

[1]Code and data available at https://github.com/yuPeiyu98/DRC

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Attempting to combine unsupervised or weakly supervised learning with modern neural networks, three lines of work surge recently for foreground extraction: (1) deep networks as feature extractors for canonical segmentation algorithms, (2) GAN-based foreground-background disentanglement, and (3) compositional latent variable models with slot-based object modeling. Despite great successes of these methods, the challenge of unsupervised foreground extraction remains largely open.

Specifically, the first line of work trains designated deep feature extractors for canonical segmentation algorithms or metric networks as learned partitioning criteria [19–21]. These methods (*e.g.*, W-Net [19]) define foreground objects' properties using learned features or criteria and are thus generally bottle-necked by the selected post-processing segmentation algorithm [22, 23]. As a branch of pioneering work that moves beyond these limitations, Yang et al. [24, 25] have recently proposed a general contextual information separation principle and an efficient adversarial learning method that is generally applicable to unsupervised segmentation, separation and detection. GAN-based models [26–31] capture the foreground objectness with oversimplified assumptions or require additional supervision to achieve foreground-background disentanglement. For example, the segmentation model in ReDO [28] is trained by redrawing detected objects, which potentially limits its application to datasets with diverse object shapes. OneGAN [31] and its predecessors [29, 30], though producing impressive results on foreground extraction, require a set of background images without foreground objects as additional inputs. Lastly, compositional latent variable models [32–40] include the background as a "virtual object" and induce the independence of object representations using an identical generator for all object slots. Although these methods exhibit strong performance on synthetic multi-object datasets with simple backgrounds and foreground shapes, they may fail on complex real-world data or even synthetic datasets with more challenging backgrounds [37, 38]. In addition, few unsupervised learning methods have provided explicit identification of foreground objects and background regions. While they can generate valid segmentation masks, most of these methods do not specify which output corresponds to the foreground objects. These deficiencies necessitate rethinking the problem of unsupervised foreground extraction. We propose to confront the challenges in formulating (1) a generic inductive bias for modeling foreground and background regions that can be baked into neural generators, and (2) an effective inference algorithm based on a principled criterion for foreground-background partition.

Inspired by Region Competition [1], a seminal approach that combines optimization-based inference [41–43] and probabilistic visual modeling [44, 45] by minimizing a generalized Bayes criterion [46], we propose to solve the foreground extraction problem by reconciling energy-based prior [47] with generative image modeling in the form of Mixture of Experts (MoE) [48, 49]. To generically describe background regions, we further introduce the learned pixel re-assignment as the essential inductive bias to capture their regularities. Fueled by our modeling, we propose to find the foreground-background partition through Expectation-Maximization (EM). Our algorithm effectively exploits the interaction between the mixture components during the partitioning process, echoing the intuition described in Region Competition [1]. We therefore coin our method Deep Region Competition (DRC). We summarize our **contributions** as follows:

1. We provide probabilistic foreground-background modeling by reconciling energy-based prior with generative image modeling in the form of MoE. With this modeling, the foreground-background partition can be naturally produced through EM. We further introduce an inductive bias, *pixel re-assignment*, to facilitate foreground-background disentanglement.
2. In experiments, we demonstrate that DRC exhibits more competitive performances on complex real-world data and challenging multi-object scenes compared with prior methods. Furthermore, we empirically show that using learned pixel re-assignment as the inductive bias helps to provide explicit identification for foreground and background regions.
3. We find that DRC can potentially generalize to novel foreground objects even from categories unseen during training, which may provide some inspiration for the study of out-of-distribution (OOD) generalization in more general unsupervised disentanglement.

2 Related Work

A typical line of methods frames unsupervised or weakly supervised foreground segmentation within a generative modeling context. Several methods build upon generative adversarial networks (GAN) [26] to perform foreground segmentation. LR-GAN [27] learns to generate background re-

gions and foreground objects separately and recursively, which simultaneously produces the foreground objects mask. ReDO (ReDrawing of Objects) [28] proposes a GAN-based object segmentation model, based on the assumption that replacing the foreground object in the image with a generated one does not alter the distribution of the training data, given that the foreground object is correctly discovered. Similarly, SEIGAN [29] learns to extract foreground objects by recombining the foreground objects with the generated background regions. FineGAN [30] hierarchically generates images (*i.e.*, first specifying the object shape and then the object texture) to disentangle the background and foreground object. Benny and Wolf [31] further hypothesize that a method solving an ensemble of unsupervised tasks altogether improves the model performance compared with the one that solves each individually. Therefore, they train a complex GAN-based model (OneGAN) to solve several tasks simultaneously, including foreground segmentation. Although LR-GAN and FineGAN do produce masks as part of their generative process, they cannot segment a given image. Despite SEIGAN and OneGAN achieving decent performance on foreground-background segmentation, these methods require a set of clean background images as additional inputs for weak supervision. ReDO captures the foreground objectness with possibly oversimplified assumptions, limiting its application to datasets with diverse object shapes.

On another front, compositional generative scene models [32–40], sharing the idea of scene decomposition stemming from DRAW [50], learn to represent foreground objects and background regions in terms of a collection of latent variables with the same representational format. These methods typically exploit the spatial mixture model for generative modeling. Specifically, IODINE [37] proposes a slot-based object representation method and models the latent space using iterative amortized inference [51]. Slot-Attention [38], as a step forward, effectively incorporates the attention mechanism into the slot-based object representation for flexible foreground object binding. Both methods use fully shared parameters among individual mixture components to entail permutation invariance of the learned multi-object representation. Alternative models such as MONet [36] and GENESIS [39] use multiple encode-decode steps for scene decomposition and foreground object extraction. Although these methods exhibit strong performance on synthetic multi-object datasets with simple background and foreground shapes, they may fail when dealing with complex real-world data or even synthetic datasets with more challenging background [37, 38].

More closely related to the classical methods, another line of work focuses on utilizing image features extracted by deep neural networks or designing energy functions based on data-driven methods to define the desired property of foreground objects. Pham et al. [52] and Silberman et al. [53] obtain impressive results when depth images are accessible in addition to conventional RGB images, while such methods are not directly applicable for data with RGB images alone. W-Net [19] extracts image features via a deep auto-encoder jointly trained by minimizing reconstruction error and normalized cut. The learned features are further processed by CRF smoothing to perform hierarchical segmentation. Kanezaki [20] proposes to employ a neural network as part of the partitioning criterion (inspired by Ulyanov et al. [54]) to minimize the chosen intra-region pixel distance for segmentation directly. Ji et al. [21] propose to use Invariant Information Clustering as the objective for segmentation, where the network is trained to be part of the learned distance. As an interesting extension, one may also consider adapting methods that automatically discover object structures [55] to foreground extraction. Though being pioneering work in image segmentation, the aforementioned methods are generally bottle-necked by the selected post-processing segmentation algorithm or require extra transformations to produce meaningful foreground segmentation masks. Yang et al. [24, 25] in their seminal work propose an information-theoretical principle and adversarial contextual model for unsupervised segmentation and detection by partitioning images into maximally independent sets, with the objective of minimizing the predictability of one set by the other sets. Additional efforts have also been devoted to weakly supervised foreground segmentation using image classification labels [56–58], bounding boxes [59, 60], or saliency maps [61–63].

3 Methodology

Foreground extraction performs a binary partition for the image $\mathbf{I}$ to extract the foreground region. Without explicit supervision, we propose to use learned pixel re-assignment as a generic inductive bias for background modeling, upon which we derive an EM-like partitioning algorithm. Compared with prior methods, our algorithm can handle images with more complex foreground shapes and background patterns, while providing explicit identification of foreground and background regions.

3.1 Preliminaries

Adopting the language of EM algorithm, we assume that for the observed sample $\mathbf{x} \in \mathbb{R}^D$, there exists $\mathbf{z} \in \mathbb{R}^d$ as its latent variables. The complete-data distribution is

$$p_\theta(\mathbf{z}, \mathbf{x}) = p_\alpha(\mathbf{z}) p_\beta(\mathbf{x}|\mathbf{z}), \tag{1}$$

where $p_\alpha(\mathbf{z})$ is the prior model with parameters α, $p_\beta(\mathbf{x}|\mathbf{z})$ is the top-down generative model with parameters β, and $\theta = (\alpha, \beta)$.

The prior model $p_\alpha(\mathbf{z})$ can be formulated as an energy-based model, which we refer to as the Latent-space Energy-Based Model (LEBM) [47] throughout the paper:

$$p_\alpha(\mathbf{z}) = \frac{1}{Z_\alpha} \exp\left(f_\alpha(\mathbf{z})\right) p_0(\mathbf{z}), \tag{2}$$

where $f_\alpha(\mathbf{z})$ can be parameterized by a neural network, Z_α is the partition function, and $p_0(\mathbf{z})$ is a reference distribution, assumed to be isotropic Gaussian prior commonly used for the generative model. The prior model in Eq. (2) can be interpreted as an energy-based correction or exponential tilting of the original prior distribution p_0.

The LEBM can be learned by Maximum Likelihood Estimation (MLE). Given a training sample $\mathbf{x}$, the learning gradient for α is derived as shown by Pang et al. [47],

$$\delta_\alpha(\mathbf{x}) = \mathbf{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right] - \mathbf{E}_{p_\alpha(\mathbf{z})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right]. \tag{3}$$

In practice, the above expectations can be approximated by Monte-Carlo average, which requires sampling from $p_\theta(\mathbf{z}|\mathbf{x})$ and $p_\alpha(\mathbf{z})$. This step can be done with stochastic gradient-based methods, such as Langevin dynamics [64] or Hamiltonian Monte Carlo [65].

An extension to LEBM is to further couple the vector representation $\mathbf{z}$ with a symbolic representation $\mathbf{y}$ [66]. Formally, $\mathbf{y}$ is a K-dimensional one-hot vector, where K is the number of possible $\mathbf{z}$ categories. Such symbol-vector duality can provide extra entries for auxiliary supervision; we will detail it in Section 3.4.

3.2 Generative Image Modeling

Mixture of Experts (MoE) for Image Generation Inspired by the regional homogenity assumption proposed by Zhu and Yuille [1], we use separate priors and generative models for foreground and background regions, indexed as α_k and β_k, $k = 1, 2$, respectively; see Fig. 1. This design leads to the form of MoE [48, 49] for image modeling, as shown below.

Let us start by considering only the i-th pixel of the observed image $\mathbf{x}$, denoted as $\mathbf{x}_i$. We use a binary one-hot random variable $\mathbf{w}_i$ to indicate whether the i-th pixel belongs to the foreground region. Formally, we have $\mathbf{w}_i = [w_{i1}, w_{i2}]$, $w_{ik} \in \{0, 1\}$ and $\sum_{k=1}^{2} w_{ik} = 1$. Let $w_{i1} = 1$ indicate that the i-th pixel $\mathbf{x}_i$ belongs to the foreground, and $w_{i2} = 1$ indicate the opposite.

We assume that the distribution of $\mathbf{w}_i$ is prior-dependent. Specifically, the mixture parameter π_{ik}, $k = 1, 2$, is defined as the output of a gating function $\pi_{ik} = p_\beta(w_{ik} = 1|\mathbf{z}) = \text{Softmax}(l_{ik})$; $l_{ik} = h_{\beta_k}(\mathbf{z}_k)$, $k = 1, 2$ are the logit scores given by the foreground and background generative models respectively; $\beta = \{\beta_1, \beta_2\}, \mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. Taken together, the joint distribution of $\mathbf{w}_i$ is

$$p_\beta(\mathbf{w}_i|\mathbf{z}) = \prod_{k=1}^{2} \pi_{ik}^{w_{ik}}. \tag{4}$$

The learned distribution of foreground and background contents are

$$p_\beta(\mathbf{x}_i|w_{ik} = 1, \mathbf{z}_k) = p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k) \sim \mathbf{N}(g_{\beta_k}(\mathbf{z}_k), \sigma^2\mathbf{I}),\ k = 1, 2 \tag{5}$$

where we assume that the generative model for region content, $p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)$, $k = 1, 2$, follows a Gaussian distribution parameterized by the generator network g_{β_k}. As in VAE, σ takes an assumed value. We follow the common practice and use a shared generator for parameterizing π_{ik} and $p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)$. We use separate branches only at the output layer to generate logits and contents.

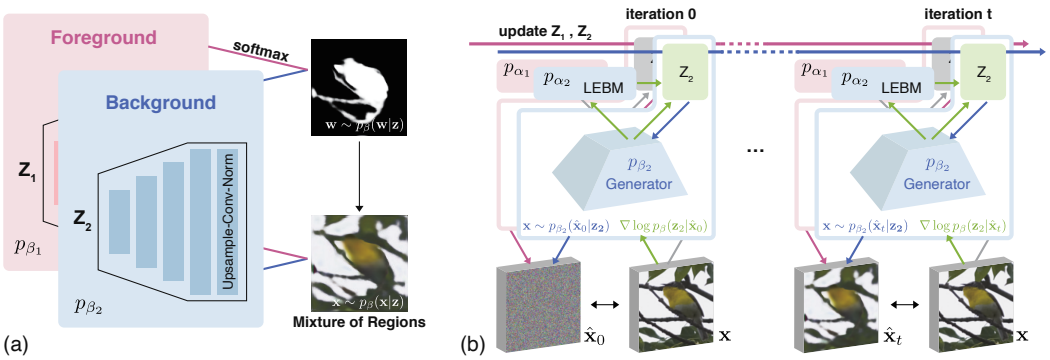

Figure 1: **Overview of DRC.** (a) The model generates foreground and background regions using sampled latent variables $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2\}$. p_{β_k}, $k = 1, 2$ represents the generator for each region. Of note, the pixel re-assignment function is absorbed in the background generator; see Section 3.2 for details. (b) DRC samples the latent variables $\mathbf{z}$ in an iterative manner. Let $\mathbf{x}$ denote the observed image; we use $\hat{\mathbf{x}}_t$, $t = 0, 1, ...$ to represent the image generated by $p_\beta(\mathbf{x}|\mathbf{z})$ at the t-th sampling step. DRC has a two-step workflow for learning unsupervised foreground extractors that resembles the E- and M-step in the classic EM algorithm. In the E-step, it employs gradient-based MCMC sampling to infer the latent variables $\mathbf{z}$ as shown in (b). Of note, only the latent variables $\mathbf{z}$ are updated in this step. In the M-step, the sampled latent variables $\mathbf{z}$ are fed into the model for image generation as shown in (a), where the generators are updated to minimize the reconstruction error.

Generating $\mathbf{x}_i$ based on $\mathbf{w}_i$'s distribution involves two steps: (1) sample $\mathbf{w}_i$ from the distribution $p_\beta(\mathbf{w}_i|\mathbf{z})$, and (2) choose either the foreground model (*i.e.*, $p_{\beta_1}(\mathbf{x}_i|\mathbf{z}_1)$) or the background model (*i.e.*, $p_{\beta_2}(\mathbf{x}_i|\mathbf{z}_2)$) to generate $\mathbf{x}_i$ based on the sampled $\mathbf{w}_i$. As such, this distribution of $\mathbf{x}_i$ is a MoE,

$$p_\beta(\mathbf{x}_i|\mathbf{z}) = \sum_{k=1}^{2} p_\beta(w_{ik} = 1|\mathbf{z}) p_\beta(\mathbf{x}_i|w_{ik} = 1, \mathbf{z}_k) = \sum_{k=1}^{2} \pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k), \tag{6}$$

wherein the posterior responsibility of w_{ik} is

$$\gamma_{ik} = p(w_{ik} = 1|\mathbf{x}_i, \mathbf{z}) = \frac{\pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)}{\sum_{m=1}^{2} \pi_{im} p_{\beta_m}(\mathbf{x}_i|\mathbf{z}_m)}, \; k = 1, 2. \tag{7}$$

Using a fully-factorized joint distribution of $\mathbf{x}$, we have $p_\beta(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^{D} \sum_{k=1}^{2} \pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)$ as the generative modeling of $\mathbf{x} \in \mathbb{R}^D$.

Learning Pixel Re-assignment for Background Modeling We use pixel re-assignment in the background generative model as the essential inductive bias for modeling the background region. This is partially inspired by the concepts of "texture" and "texton" by Julez [45, 67], where the textural part of an image may contain fewer structural elements in preattentive vision, which coincides with our intuitive observation of the background regions.

We use a separate pair of energy-based prior model $\alpha_{\texttt{pix}}$ and generative model $\beta_{\texttt{pix}}$ to learn the re-assignment. For simplicity, we absorb $\alpha_{\texttt{pix}}$ and $\beta_{\texttt{pix}}$ in the models for background modeling, *i.e.*, α_2 and β_2, respectively. In practice, the re-assignment follows the output of $\beta_{\texttt{pix}}$, a shuffling grid with the same size of the image $\mathbf{x}$. Its values indicate the re-assigned pixel coordinates; see Fig. 2. We find that shuffling the background pixels using the learned re-assignment facilitates the model to capture the regularities of the background regions. Specifically, the proposed model with this essential inductive bias learns to constantly give the correct mask assignment, whereas most previous fully unsupervised methods do not provide explicit identification of the foreground and background regions; see discussion in Section 4.1 for more details.

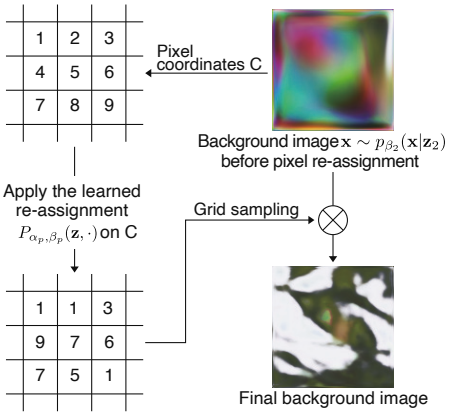

Figure 2: **Pixel re-assignment.** The output of β_p can be viewed as a learned re-assignment of the original background pixels that follows the mapped grid $P_{\alpha_p,\beta_p}(\mathbf{z}, C)$. Note that the re-assignment function $P_{\alpha_p,\beta_p}(\mathbf{z}, \cdot)$ might not be injective. The final background image is generated via grid sampling.

3.3 Deep Region Competition: from Generative Modeling to Foreground Extraction

The complete-data distribution from the image modeling is

$$\begin{aligned} p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w}) &= p_\beta(\mathbf{x}|\mathbf{w}, \mathbf{z}) p_\beta(\mathbf{w}|\mathbf{z}) p_\alpha(\mathbf{z}) \\ &= \left(\prod_{i=1}^{D} \prod_{k=1}^{2} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)^{w_{ik}} \right) \left(\prod_{i=1}^{D} \prod_{k=1}^{2} \pi_{ik}^{w_{ik}} \right) p_\alpha(\mathbf{z}) \\ &= p_\alpha(\mathbf{z}) \prod_{i=1}^{D} \prod_{k=1}^{2} \left(\pi_{ik} p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k) \right)^{w_{ik}}, \end{aligned} \tag{8}$$

where $p_\alpha(\mathbf{z}) = p_{\alpha_1}(\mathbf{z}_1) p_{\alpha_2}(\mathbf{z}_2)$ is the prior model given by LEBMs. $\alpha = \{\alpha_1, \alpha_2\}$, and $\theta = \{\alpha, \beta\}$. $\mathbf{w}$ is the vector of $(\mathbf{w}_i), i = 1, ...D$, whose joint distribution is assumed to be fully-factorized.

Next, we derive the complete-data log-likelihood as our learning objective:

$$\mathcal{L}(\theta) = \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w}) = \log p_\alpha(\mathbf{z}) + \sum_{i=1}^{D} \sum_{k=1}^{2} w_{ik} \left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k) \right). \tag{9}$$

Of note, $\mathbf{w}$ and $\mathbf{z}$ are unobserved variables in the modeling, which makes it impossible to learn the model directly through MLE. To calculate the gradients of θ, we instead optimize $\mathbf{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x}), \mathbf{w}\sim p(\mathbf{w}|\mathbf{x},\mathbf{z})}[\mathcal{L}(\theta)]$ based on the fact that underlies the EM algorithm:

$$\begin{aligned} \nabla_\theta \log p_\theta(\mathbf{x}) &= \int_{\mathbf{z}} p_\theta(\mathbf{z}|\mathbf{x}) d\mathbf{z} \int_{\mathbf{w}} p_\theta(\mathbf{w}|\mathbf{z}, \mathbf{x}) \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w}) d\mathbf{w} \\ &= \mathbf{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x}), \mathbf{w}\sim p_\theta(\mathbf{w}|\mathbf{x},\mathbf{z})}[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{w})]. \end{aligned} \tag{10}$$

Therefore, the derived surrogate learning objective becomes

$$\max_\theta \ \mathbf{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})} \left[\mathcal{J}(\theta) \right], \ \text{s.t.} \ \forall i, \sum_{k=1}^{2} \pi_{ik} = 1, \tag{11}$$

$$\mathcal{J}(\theta) = \underbrace{\log p_\alpha(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D} \sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D} \sum_{k=1}^{2} \gamma_{ik} \log p_{\theta_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}, \tag{12}$$

where $\mathcal{J}(\theta) = \mathbf{E}_{\mathbf{w}\sim p_\theta(\mathbf{w}|\mathbf{x},\mathbf{z})} \left[\mathcal{L}(\theta) \right]$ is the conditional expectation of $\mathbf{w}$, which can be calculated in closed form; see the supplementary material for additional details.

Eq. (11) has an intuitive interpretation. We can decompose the learning objective into three components as in Eq. (12). In particular, the second term $\sum_{i=1}^{D} \sum_{k=1}^{2} \gamma_{ik} \log \pi_{ik}$ has a similar form to the cross-entropy loss commonly used for supervised segmentation task, where the posterior responsibility γ_{ik} serves as the target distribution. It is as if the foreground and background generative models compete with each other to fit the distribution of each pixel $\mathbf{x}_i$. If the pixel value at $\mathbf{x}_i$ fits better to the distribution of foreground, $p_{\beta_1}(\mathbf{x}_i|\mathbf{z}_1)$, than to that of background, $p_{\beta_2}(\mathbf{x}_i|\mathbf{z}_2)$, the model tends to assign that pixel to the foreground region (see Eq. (7)), and vice versa. This mechanism is similar to the process derived in Zhu and Yuille [1], which is the reason why we coin our method Deep Region Competition (DRC).

Prior to our proposal, several methods [1, 37, 38] also employ mixture models and competition among the components to perform unsupervised foreground or image segmentation. The original Region Competition [1] combines several families of image modeling with Bayesian inference but is limited by the expressiveness of the pre-specified probability distributions. More recent methods, including IODINE [37] and Slot-attention [38], learn amortized inference networks for latent variables and induce the independence of foreground and background representations using an identical generator. Our method combines the best of the two worlds, reconciling the expressiveness of learned generators with the regularity of generic texture modeling under the framework of LEBM.

To optimize the learning objective in Eq. (11), we approximate the expectation by sampling from the prior $p_\alpha(\mathbf{z})$ and posterior model $p_\theta(\mathbf{z}|\mathbf{x}) \propto p_\alpha(\mathbf{z}) p_\beta(\mathbf{x}|\mathbf{z})$, followed by calculating the Monte Carlo average. We use Langevin dynamics [64] to draw persistent MCMC samples, which iterates

$$\mathbf{z}_{t+1} = \mathbf{z}_t + s \nabla_{\mathbf{z}} \log Q(\mathbf{z}_t) + \sqrt{2s}\epsilon_t, \tag{13}$$

where t is the Langevin dynamics's time step, s the step size, and ϵ_t the Gaussian noise. $Q(\mathbf{z})$ is the target distribution, being either $p_\alpha(\mathbf{z})$ or $p_\theta(\mathbf{z}|\mathbf{x})$. $\nabla_{\mathbf{z}} \log Q(\mathbf{z}_t)$ is efficiently computed via automatic differentiation in modern learning libraries [68]. We summarize the above process in Algorithm 1.

Algorithm 1: Learning models of DRC via EM.

Input: Learning iterations T, initial parameters for LEBMs $\alpha^{(0)} = \{\alpha_1^{(0)}, \alpha_2^{(0)}\}$ and generators $\beta^{(0)} = \{\beta_1^{(0)}, \beta_2^{(0)}\}$, $\theta^{(0)} = \{\alpha^{(0)}, \beta^{(0)}\}$, learning rate η_α for LEBMs, η_β for foreground and background generators, observed examples $\{\mathbf{x}^{(i)}\}_{i=1}^N$, batch size M, and initial latent variables $\{\mathbf{z}_-^{(i)} = \{\mathbf{z}_{1-}^{(i)}, \mathbf{z}_{2-}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$ and $\{\mathbf{z}_+^{(i)} = \{\mathbf{z}_{1+}^{(i)}, \mathbf{z}_{2+}^{(i)}\} \sim p_0(\mathbf{z})\}_{i=1}^N$.

Output: $\theta^{(T)} = \{\alpha_1^{(T)}, \beta_1^{(T)}, \alpha_2^{(T)}, \beta_2^{(T)}\}$.

for $t = 0 : T-1$ **do**

- Sample a minibatch of data $\{\mathbf{x}^{(i)}\}_{i=1}^M$;
- **Prior sampling for learning LEBMs:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_-^{(i)}$ using Eq. (13), with target distribution $\pi(\mathbf{z}) = p_{\alpha^{(t)}}(\mathbf{z})$;
- **Posterior sampling for foreground and background generation:** For each $\mathbf{x}^{(i)}$, update $\mathbf{z}_+^{(i)}$ using Eq. (13), with target distribution $Q(\mathbf{z}) = p_{\theta^{(t)}}(\mathbf{z}|\mathbf{x})$;
- **Update LEBMs:** $\alpha^{(t+1)} = \alpha^{(t)} + \eta_\alpha \frac{1}{m} \sum_{i=1}^m [\nabla_\alpha f_{\alpha^{(t)}}(\mathbf{z}_+^{(i)}) - \nabla_\alpha f_{\alpha^{(t)}}(\mathbf{z}_-^{(i)})]$;
- **Update foreground and background generators:** $\beta^{(t+1)} = \beta^{(t)} + \eta_\beta \frac{1}{m} \sum_{i=1}^m \nabla_\beta \log p_{\beta^{(t)}}(\mathbf{x}^{(i)}|\mathbf{z}_+^{(i)})$;

During inference, we initialize the latent variables $\mathbf{z}$ for MCMC sampling from Gaussian white noise and run only the posterior sampling step to obtain $\mathbf{z}_+$. The inferred mask and region images are then given by the outputs of generative models $p_{\beta_k}(\mathbf{w}|\mathbf{z}_+)$ and $p_{\beta_k}(\mathbf{x}|\mathbf{z}_+),\ k = 1, 2$, respectively.

3.4 Technical Details

Pseudo label for additional regularization Although the proposed DRC explicitly models the interaction between the regions, it is still possible that the model converges to a trivial extractor, which treats the entire image as the foreground or background region, leaving the other region null. We exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM (see Section 3.1) for additional regularization. The strategy is similar to the mutual information maximization used in InfoGAN [69]. Specifically, we use the symbolic vector $\mathbf{y}$ inferred from $\mathbf{z}$ as the pseudo-class label for $\mathbf{z}$ and train an auxiliary classifier jointly with the above models; it ensures that the generated regions $\mathbf{x}_k$ contain similar symbolic information for $\mathbf{z}_k$. Intuitively, this loss prevents the regions from converging to null since the symbolic representation $\mathbf{y}_k$ would never be well retrieved if that did happen.

Implementation We adopt a similar architecture for the generator as in DCGAN [70] throughout the experiments and only change the dimension of the latent variables $\mathbf{z}$ for different datasets. The generator consists of a fully connected layer followed by five stacked upsample-conv-norm layers. We replace the batch-norm layers [71] with instance-norm [72] in the architecture. The energy-term in LEBM is parameterized by a 3-layered MLP. We adopt orthogonal initialization [73] commonly used in generative models to initialize the networks and orthogonal regularization [74] to facilitate training. In addition, we observe performance improvement when adding Total-Variation norm [75] for the background generative model. More details, along with specifics of the implementation used in our experiments, are provided in the supplementary material.

4 Experiments

We design experiments to answer three questions: (1) How does the proposed method compare to previous state-of-the-art competitors? (2) How do the proposed components contribute to the model performance? (3) Does the proposed method exhibit generalization on images containing unseen instances (*i.e.*, same category but not the same instance) and even objects from novel categories?

To answer these questions, we evaluate our method on five challenging datasets in two groups: (1) Caltech-UCSD Birds-200-2011 (Birds) [76], Stanford Dogs (Dogs) [77], and Stanford Cars (Cars) [78] datasets; (2) CLEVR6 [79] and Textured Multi-dSprites (TM-dSprites) [80] datasets. The first group of datasets covers complex real-world domains, whereas the second group features environments of the multi-object foreground with challenging spatial configurations or confounding backgrounds. As to be shown, the proposed method is generic to various kinds of input and produces more competitive foreground-background partition results than prior methods.

4.1 Results on Foreground Extraction

Model	Single Object						Multi-Object			
	Birds		Dogs		Cars		CLEVR6		TM-dSprites	
	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
W-Net*	24.8	38.9	47.7	62.1	52.8	67.6	-	-	-	-
GrabCut	30.2	42.7	58.3	70.9	61.3	73.1	19.0	30.5	61.9	71.0
ReDO§	46.5	60.2	55.7	70.3	52.5	68.6	18.6	31.0	9.4	17.2
OneGAN*†	55.5	69.2	71.0	81.7	71.2	82.6	-	-	-	-
IODINE§	30.9	44.6	54.4	67.0	51.7	67.3	19.9	32.4	7.3	12.8
Slot-Attn.§	35.6	51.5	38.6	55.3	41.3	58.3	83.6	90.7	7.3	13.5
Ours	**56.4**	**70.9**	**71.7**	**83.2**	**72.4**	**83.7**	**84.7**	**91.5**	**78.8**	**87.5**

Table 1: **Foreground extraction results on training data measured in IoU and Dice.** Higher is better in all scores. *Results of W-Net and OneGAN are provided by Benny and Wolf [31]. Of note, results of these two models on Dogs and Cars datasets may **not** be directly comparable to other listed methods, as the data used for training and evaluation could be different. We include these results as a rough reference since no official implementation or pretrained model are publicly available. § indicates unfair baseline results obtained using extra ground-truth information, *i.e.*, we choose the best-matching scores from the permutation of foreground and background masks. †OneGAN is a strong **weakly supervised** baseline, which requires clean background images to provide additional supervision. We include this model as a potential upper bound of the fully unsupervised methods.

Single object in the wild In the first group of datasets, there is typically a single object in the foreground, varying in shapes, texture, and lighting conditions. Unsupervised foreground extraction on these datasets requires much more sophisticated visual cues than colors and shapes. Birds dataset consists of 11,788 images of 200 classes of birds annotated with high-quality segmentation masks, Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes, and Cars dataset consists of 16,185 images of 196 classes. The latter two datasets are primarily made for fine-grained categorization. To evaluate foreground extraction, we follow the practice in Benny and Wolf [31], and approximate ground-truth masks for the images with Mask R-CNN [16], pre-trained on the MS COCO [9] dataset with a ResNet-101 [81] backend. The pre-trained model is acquired from the detectron2 [82] toolkit. This results in 5,024 dog images and 12,322 car images with a clear foreground-background setup and corresponding masks.

On datasets featuring a single foreground object, we use the 2-slot version of IODINE and Slot-attention. Since ReDO, IODINE, and Slot-Attention do not distinguish foreground and background in output regions, we choose the best-matching scores from the permutation of foreground and background masks as in [28]. We observe that the proposed method and Grabcut are the only two methods that provide explicit identification of foreground objects and background regions. While the Grabcut algorithm actually requires a predefined bounding box as input that specifies the foreground region, our method, thanks to the learned pixel re-assignment (see Section 3.2), can achieve this in a fully unsupervised manner. Results in Table 1 show that our method outperforms all the unsupervised baselines by a large margin, exhibiting comparable performance even to the weakly supervised baseline that requires additional background information as inputs [31]. We provide samples of foreground extraction results as well as generated background and foreground regions in Fig. 3. Note that our final goal is not to synthesize appealing images but to learn foreground extractors in a fully unsupervised manner. As the limitation of our method, DRC generates foreground and background regions less realistic than those generated by state-of-the-art GANs, which hints a possible direction for future work. More detailed discussions of the limitation can be found in supplementary material.

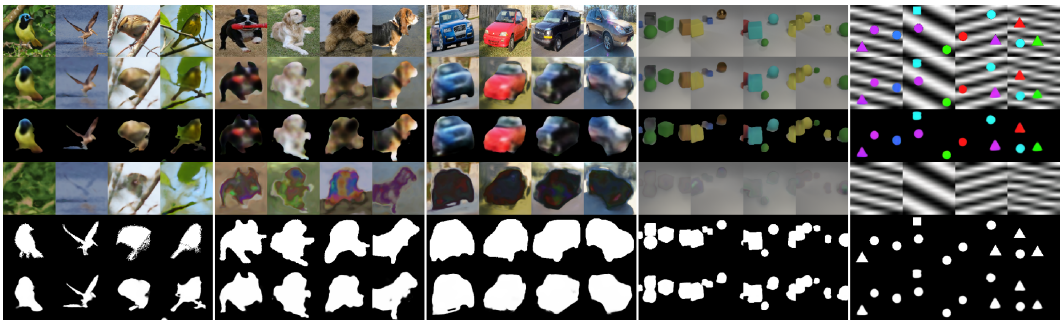

Figure 3: **Foreground extraction results for each dataset**; zoom in for better visibility. From top to bottom: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks. More samples and results of baselines can be found in the supplementary material.

Multi-object scenes The second group of datasets contains images with possibly simpler foreground objects but more challenging scene configurations or background parts. Visual scenes in the CLEVR6 dataset contain various objects and often with partial occlusions and truncations. Following the evaluation protocol in IODINE and Slot-attention, we use the first 70K samples from CLEVR [79] and filter the samples for scenes with at most 6 objects for training and evaluation, *i.e.*, CLEVR6. The TM-dSprites dataset is a variant of Multi-dSprites [80] but has strongly confounding backgrounds borrowed from Textured MNIST [32]. We generate 20K samples for the experiments. Similar to Greff et al. [37] and Locatello et al. [38], we evaluate on a subset containing 1K samples for testing. Note that IODINE and Slot-attention are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the background is viewed as a virtual "object" as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned in Locatello et al. [38]. We therefore propose two corresponding approaches (see the supplementary material for more details) to convert the output object masks into a foreground-background partition and report the best results of these two options for IODINE and Slot-attention in Table 1.

On the CLEVR6 dataset, we use the publicly available pretrained model for IODINE, which achieves a reasonable ARI (excluding background pixels) of 94.4 on the testing data, close to the testing results in Greff et al. [37]. We observe that IODINE distributes the background over all the slots for some of the testing samples, resulting in much lower IoU and Dice scores. We re-train the Slot-attention model using the official implementation on CLEVR6, as no pretrained model is publicly available. The re-trained model achieves a foreground ARI of 98.0 on the 1K testing samples, which we consider as a sign of valid re-implementation. Results in Table 1 demonstrate that the proposed method can effectively process images of challenging multi-object scenes. To be specific, our method demonstrates competitive performance on the CLEVR6 dataset compared with the SOTA object discovery method. Moreover, as shown empirically in Fig. 3, the proposed method can handle the strongly confounding background introduced in Greff et al. [32], whereas previous methods are distracted by the background and mostly fail to capture the foreground objects.

4.2 Ablation Study

We provide ablation studies on the Birds dataset to inspect the contribution of each proposed component in our model. As shown in Table 2, we observe that replacing the LEBMs in the foreground and background models with amortized inference networks significantly harms the performance of the proposed method. In particular, the modified model fails to generate any meaningful results (marked as - in Table 2). We conjecture that LEBM benefits from the low-dimensionality of the latent space [47] and therefore enjoys more efficient learning. However, the inference networks need to learn an extra mapping from the high-dimensional image space to the latent space and require more elaborate architecture and tuning for convergence. Furthermore, we observe that the

Model	IoU	Dice
amortized inference*	-	-
w/o pix. re-assign.	21.8	35.3
w/o pseudo label	48.7	64.2
w/o TV-norm reg.	53.0	68.1
w/o ortho. reg.	54.3	69.2
short-run chain†	52.5	67.7
Full model	56.4	70.9

Table 2: **Ablation study on Birds.** *We replace the LEBM with encoders to perform amortized inference for the latent variables $\mathbf{z}$ within a variational framework as in VAE [83]. †We explore the possibility of using short-run MCMC [84] instead of persistent chain sampling.

model that does not learn pixel re-assignment for background can still generate meaningful images but only vaguely captures masks for foreground extraction.

4.3 Generalizable Foreground Extraction

Extracting novel foreground objects from training categories We show results on generalizing to novel objects from the training classes. To evaluate our method, we split the Birds dataset following Chen et al. [28], resulting in 10K training images and 1K testing images. On Dogs and Cars datasets, we split the dataset based on the original train-test split [77, 78]. This split gives 3,286 dog images and 6,218 car images for training, and 1,738 dog images and 6,104 car images for testing, respectively. As summarized in Table 3, our method shows superior performances compared with baselines; the performance gap between training and testing is constantly small over all datasets.

Model	Birds		Dogs		Cars	
	IoU	Dice	IoU	Dice	IoU	Dice
	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.	Tr.|Te.
GrabCut*	30.2|30.3	42.7|42.8	58.3|57.9	70.8|70.5	60.9|61.6	72.7|73.5
ReDO	46.8|47.1	61.4|61.7	54.3|52.8	69.2|67.9	52.6|52.5	68.7|68.6
Ours	**54.8|54.6**	**69.5|69.4**	**71.6|72.3**	**83.2|83.6**	**71.9|70.8**	**83.3|82.5**

Table 3: **Performance of DRC on training and held-out testing data.** *Note that GrabCut is a deterministic method that does not require training. We report the results of GrabCut [5] on these splits only for reference. Tr. indicates the performance on training data, and Te. indicates the performance on testing data.

Extracting novel foreground objects from unseen categories To investigate how well our method generalizes to categories unseen during training, we evaluate the models trained in real-world single object datasets on the held-out testing data from different categories. We use the same training and testing splits on these datasets as in the previous experiments. Table 4 shows that our method outperforms the baselines on the Birds dataset when the model has trained on Dogs or Cars dataset, which have quite different distributions in foreground object shapes. Competitors like ReDO also exhibit such out-of-distribution generalization but only to a limited extent. Similar results are observed when using Dogs or Cars as the testing set. We can see that when the model is trained on Dogs and evaluated on Cars or vice versa, it still maintains comparable performances w.r.t. those are trained&tested on the same class. We hypothesize that these two datasets have similar distributions in foreground objects and background regions. In the light of this, we can further entail that the distribution of Dogs is most similar to that of Cars and less similar to that of Birds according to the results, which is consistent with our intuitive observation of the data. We provide a preliminary analysis of the statistics of these datasets in the supplementary material.

Test	Train	GrabCut		ReDO		Ours	
		IoU	Dice	IoU	Dice	IoU	Dice
Birds	Birds*	30.3	42.8	47.1	61.7	54.6	69.4
	Dogs			22.2	35.3	**41.3**	**57.4**
	Cars			16.4	27.7	39.2	55.3
Dogs	Dogs*	57.9	70.5	52.8	67.9	72.3	83.6
	Cars			44.5	61.2	**67.8**	**80.4**
	Birds			44.0	60.3	53.6	69.1
Cars	Cars*	61.6	73.5	52.5	68.6	70.8	82.5
	Dogs			51.6	67.1	**68.6**	**81.0**
	Birds			41.8	58.6	45.1	61.7

Table 4: **Performance of DRC on unseen testing categories.** *We include the testing results of models trained with data from the same categories for reference.

5 Conclusion

We have presented the Deep Region Competition, an EM-based fully unsupervised foreground extraction algorithm fueled by energy-based prior and generative image modeling. We propose learned pixel re-assignment as an inductive bias to capture the background regularities. Experiments demonstrate that DRC exhibits more competitive performances on complex real-world data and challenging multi-object scenes. We show empirically that learned pixel re-assignment helps to provide explicit identification for foreground and background regions. Moreover, we find that DRC can potentially generalize to novel foreground objects even from categories unseen during training. We hope our work will inspire future research along this challenging but rewarding research direction.

Acknowledgements

The work was supported by NSF DMS-2015577, ONR MURI project N00014-16-1-2007, and DARPA XAI project N66001-17-2-4029. We would like to thank Bo Pang from the UCLA Statistics Department for his insights on the latent-space energy-based model and four anonymous reviewers (especially Reviewer RcgA) for their constructive comments.

References

[1] Song-Chun Zhu and Alan Yuille. Region competition: Unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(9):884–900, 1996.

[2] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.

[3] Zhuowen Tu and Song-Chun Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):657–673, 2002.

[4] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2001.

[5] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

[6] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37 (3):569–582, 2014.

[7] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[8] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision (IJCV)*, 88(2): 303–338, 2010.

[11] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017.

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, 2015.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2017.

[17] Nick Chater, Joshua B Tenenbaum, and Alan Yuille. Probabilistic models of cognition: Conceptual foundations. *Trends in Cognitive Sciences*, 10(7):287–291, 2006.

[18] Thomas F Shipley and Philip J Kellman. *From fragments to objects: Segmentation and grouping in vision*. Elsevier, 2001.

[19] Xide Xia and Brian Kulis. W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*, 2017.

[20] Asako Kanezaki. Unsupervised image segmentation by backpropagation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[21] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[22] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5): 898–916, 2010.

[23] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(11):2274–2282, 2012.

[24] Yanchao Yang, Antonio Loquercio, Davide Scaramuzza, and Stefano Soatto. Unsupervised moving object detection via contextual information separation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Yanchao Yang, Brian Lai, and Stefano Soatto. Dystab: Unsupervised object segmentation via dynamic-static bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[26] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[27] Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. Lr-gan: Layered recursive generative adversarial networks for image generation. In *International Conference on Learning Representations (ICLR)*, 2017.

[28] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[29] Pavel Ostyakov, Roman Suvorov, Elizaveta Logacheva, Oleg Khomenko, and Sergey I Nikolenko. Seigan: Towards compositional image generation by simultaneously learning to segment, enhance, and inpaint. *arXiv preprint arXiv:1811.07630*, 2018.

[30] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[31] Yaniv Benny and Lior Wolf. Onegan: Simultaneous unsupervised learning of conditional image generation, foreground segmentation, and fine-grained clustering. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[32] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[33] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[34] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[35] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *International Conference on Learning Representations (ICLR)*, 2018.

[36] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[37] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[38] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[39] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*, 2020.

[40] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*, 2020.

[41] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331, 1988.

[42] Laurent D Cohen. On active contour models and balloons. *CVGIP: Image understanding*, 53(2):211–218, 1991.

[43] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(6):641–647, 1994.

[44] Song-Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *International Journal of Computer Vision (IJCV)*, 27(2):107–126, 1998.

[45] Cheng-en Guo, Song-Chun Zhu, and Ying Nian Wu. Primal sketch: Integrating structure and texture. *Computer Vision and Image Understanding (CVIU)*, 106(1):5–19, 2007.

[46] Yvan G Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision (IJCV)*, 3(1):73–102, 1989.

[47] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[48] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[49] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[50] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

[51] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

[52] Trung T Pham, Thanh-Toan Do, Niko Sünderhauf, and Ian Reid. Scenecut: Joint geometric and object segmentation for indoor scenes. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

[53] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.

[54] D Ulyanov, A Vedaldi, and V Lempitsky. Deep image prior. *International Journal of Computer Vision (IJCV)*, 128(7), 2020.

[55] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Bjorn Ommer. Unsupervised part-based disentangling of object shape and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[56] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[57] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[58] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. Weakly-supervised semantic segmentation network with deep seeded region growing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[59] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[60] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] Seong Joon Oh, Rodrigo Benenson, Anna Khoreva, Zeynep Akata, Mario Fritz, and Bernt Schiele. Exploiting saliency for object segmentation from image level labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[62] Yu Zeng, Yunzhi Zhuge, Huchuan Lu, and Lihe Zhang. Joint learning of saliency detection and weakly supervised semantic segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[63] Andrey Voynov, Stanislav Morozov, and Artem Babenko. Object segmentation without labels with large-scale generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[64] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of International Conference on Machine Learning (ICML)*, 2011.

[65] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[66] Bo Pang and Ying Nian Wu. Latent space energy-based model of symbol-vector coupling for text generation and classification. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

[67] Bela Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[69] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[70] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[71] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning (ICML)*, 2015.

[72] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[73] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[74] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[75] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[76] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[77] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

[78] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, 2013.

[79] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[80] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[82] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[83] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[84] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run mcmc toward energy-based model. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[85] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 4.1, Section 4.2 and Section 4.3.
   (b) Did you describe the limitations of your work? [Yes] See Section 4.1.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide pytorch-style codes and detailed instructions of how to reproduce the main results in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1, Section 4.2, Section 4.3 and supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4.1, Section 4.2 and Section 4.3.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Dataset Details

A.1 Caltech-UCSD Birds-200-2011

Birds dataset consists of 11,788 images of 200 classes of birds annotated with high-quality segmentation masks. Each image is further annotated with 15 part locations, 312 binary attributes, and 1 bounding box. We use the provided bounding box to extract a center square from the image, and scale it to 128×128 pixels. Each scene contains exactly one foreground object.

A.2 Stanford Dogs

Dogs dataset consists of 20,580 images of 120 classes annotated with bounding boxes. We first use the provided bounding box to extract the center square, and then scale it to 128×128 pixels. As stated in the paper, we approximate ground-truth masks for the pre-processed images with Mask R-CNN [16], pre-trained on the MS COCO [9] dataset with a ResNet-101 [81] backend. The pretrained model is acquired from the detectron2 [82] toolkit. We exclude the images where no dog is detected. We then manually exclude those images where the foreground object has occupied more than $\sim 90\%$ of the image, those with poor masks, and those with significant foreground distractors such as humans (see Fig. S1). The filtering strategy results in 5,024 images with a clear foreground-background setup and high-quality mask.

Figure S1: Examples of excluded images. From left to right: (i) image with a foreground object that occupied too much space, (ii) image with a low-quality mask, and (iii) image with significant foreground distractors.

A.3 Stanford Cars

Cars dataset consists of 16,185 images of 196 classes annotated with bounding boxes. Though also being primarily designed for fine-grained categorization, it has a much clearer foreground-background setup compared with the Dogs dataset. We employ a similar process as used for Dogs dataset to approximate the ground-truth masks, and only exclude those images where cars are not properly detected. It finally produces 12,322 images for our experiments.

A.4 CLEVR6

CLEVR6 dataset is a subset of the original CLEVR dataset [79] with masks, generated by Greff et al. [37]. We follow the evaluation protocol adopted by IODINE [37] and Slot-attention [38], which takes the first 70K samples from CLEVR. These samples are then filtered to only include scenes with at most 6 objects. Additionally, we perform a center square crop of 192×192 from the original 240×320 image, and scale it to 128×128 pixels. The resulting CLEVR6 dataset contains 3-6 foreground objects that could be with partial occlusion and truncation in each visual scene.

A.5 Textured Multi-dSprites

TM-dSprites dataset, which is based on the dSprites dataset [80] and Textured MNIST [32], consists of 20,000 images with a resolution of 128×128. Each image contains 2-3 random sprites, which vary in terms of shape (square, circle, or triangle), color (uniform saturated colors), and position (continuous). The background regions are borrowed from Textured MNIST dataset [32]. The textures for the background are randomly shifted samples from a bank of 20 sinusoidal textures with different frequencies and orientations. We adopt a simpler foreground setting compared with the vanilla Multi-dSprites dataset used in [37], *i.e.*, the foreground objects are not occluded as the dataset is designed to emphasize the background part. Some samples are presented in Fig. S2.

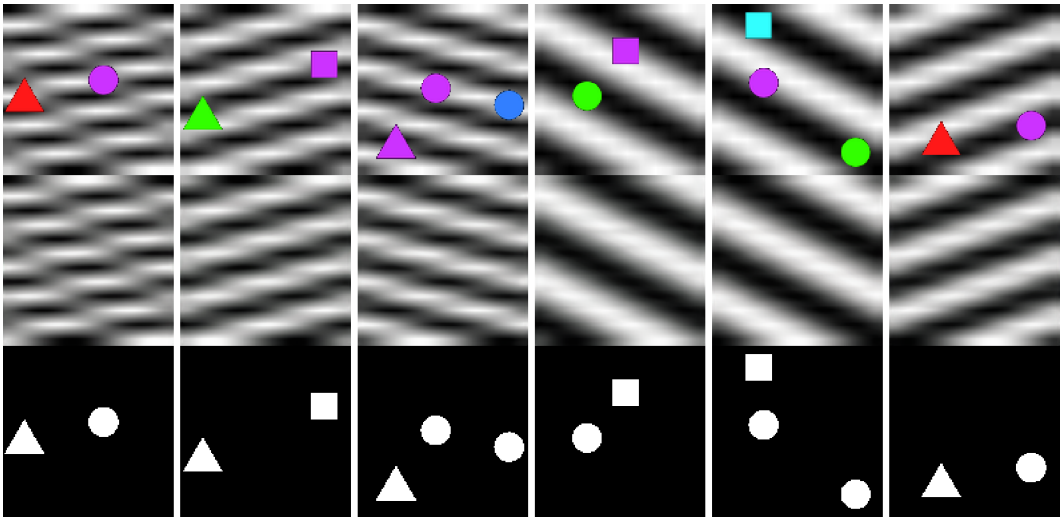

Figure S2: Samples from TM-dSprites. From top to bottom: (1) observed images, (ii) background textures, and (iii) ground-truth masks.

B Details on Models and Hyperparameters

Architecture As mentioned in the paper, we use the same overall architecture for different datasets (while the size of latent variables may vary). The details for the generators and LEBMs are summarized in the Table S1 and Table S2.

Dataset	Foreground	Background	Pixel Re-assignment
Birds	256	256	512
Dogs	256	256	512
Cars	256	192	512
CLEVR6	256	2	256
TM-dSprites	256	4	1024

Table S1: Dimension of latent variables on each dataset.

Hyperparameters and Training Details For the Langevin dynamics sampling [64], we use K_0 and K_1 to denote the number of prior and posterior sampling steps with step sizes s_0 and s_1 respectively. Our hyperparameter choices are: $K_0 = 60, K_1 = 40, s_0 = 0.4$ and $s_1 = 0.1$. These are identical across different datasets. During testing, we set the posterior sampling steps to 300 for Dogs and Cars, and 2.5K, 5K and 5K for Birds, CLEVR6 and TM-dSprites respectively. The parameters of the generators and LEBMs are initialized with orthogonal initialization [73]. The gain is set to 1.0 for all the models. We use the ADAM optimizer [85] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Generators are trained with a constant learning rate of 0.0001, and LEBMs with 0.00002. We run experiments on a single V100 GPU with 16GB of RAM and with a batch size of 48. We set the maximum training iterations to 10K and run for at most 48hrs for each dataset.

Layers	In-Out size	Comment
LEBM for Foreground/Background Models		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	$K^\dagger$	
LEBM for Pixel Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	200	
Linear, LReLU	200	
Linear, LReLU	200	
Linear	1	
Generator for Foreground/Background Model and Re-assignment Model		
Input: $\mathbf{z}$	D^*	
Linear, LReLU	$4 \times 4 \times 128$	reshaped output
UpConv3x3Norm, LReLU	$8 \times 8 \times 1024$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$16 \times 16 \times 512$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$32 \times 32 \times 256$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$64 \times 64 \times 128$	stride 1 & padding 1
UpConv3x3Norm, LReLU	$128 \times 128 \times 64$	stride 1 & padding 1
Conv3x3	$128 \times 128 \times (3+1)$	RGB & Mask
	$128 \times 128 \times 2$	Re-assignment grid
Auxiliary classifier for Foreground/Background Model		
Input: $\mathbf{x}$	$128 \times 128 \times 3$	generated image
Conv4x4Norm, LReLU	$64 \times 64 \times 64$	stride 2 & padding 1
Conv4x4Norm, LReLU	$32 \times 32 \times 128$	stride 2 & padding 1
Conv4x4Norm, LReLU	$16 \times 16 \times 256$	stride 2 & padding 1
Conv4x4Norm, LReLU	$8 \times 8 \times 512$	stride 2 & padding 1
Conv4x4Norm, LReLU	$4 \times 4 \times 1024$	stride 2 & padding 1
Conv4x4	$1 \times 1 \times K^\dagger$	

Table S2: Architecture of the generators, LEBMs and auxiliary classifiers (see Appendix C.2). UpConv3x3Norm denotes a Upsampling-Convolutional-InstanceNorm layer with a convolution kernel size of 3. Similarly, Conv4x4Norm denotes a Convolutional-InstanceNorm layer with a kernel size of 4. LReLU denotes the Leaky-ReLU activation function. The leak factor for LReLU is 0.2 in LEBMs and auxiliary classifiers, and 0.01 in generators. $*D$ represents the dimensions of the latent variables for different datasets; see Table S1. $\dagger K$ represents the pre-specified category number for latent variables. We use 200 for both the foreground and background LEBMs on real-world datasets, and 30 and 10 in the foreground and background LEBMs on multi-object datasets respectively.

C Details on Learning Objective and Regularization

C.1 Learning Objective

Derivation of Surrogate Learning Objective $\mathcal{J}(\theta) = \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}[\mathcal{L}(\theta)]$ is the conditional expectation of $\mathbf{w}$,

$$\begin{aligned}\mathcal{J}(\theta) &= \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}[\mathcal{L}(\theta)] \\ &= \log p_\alpha(\mathbf{z}) + \mathbf{E}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right)\right] \\ &= \log p_\alpha(\mathbf{z}) + \sum_{i=1}^{D}\sum_{k=1}^{2} \mathbf{E}[w_{ik}]\left(\log \pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)\right),\end{aligned} \tag{14}$$

where $\mathbf{E}$ is the conditional expectation of $\mathbf{w}$. Recall that $w_{ik} \in \{0, 1\}$. The expectation becomes

$$\begin{aligned}\mathbf{E}[w_{ik}] &= 0 \times p(w_{ik} = 0|\mathbf{x}_i, \mathbf{z}) + 1 \times p(w_{ik} = 1|\mathbf{x}_i, \mathbf{z}) \\ &= \gamma_{ik},\end{aligned} \tag{15}$$

which is the posterior responsibility of w_{ik}. We can further decompose $\mathcal{J}(\theta)$ into

$$\mathcal{J}(\theta) = \underbrace{\log p_\alpha(\mathbf{z})}_{\text{objective for LEBM}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik}\log\pi_{ik}}_{\text{foreground-background partitioning}} + \underbrace{\sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik}\log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_k)}_{\text{objective for image generation}}, \tag{16}$$

as mentioned in the paper.

Understanding the Optimization Process Note that the surrogate learning objective is an expectation w.r.t $\mathbf{z}$,

$$\max_\theta \ \mathbf{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x})}\left[\mathcal{J}(\theta)\right], \text{ s.t. } \forall i, \sum_{k=1}^{2}\pi_{ik} = 1, \tag{17}$$

which is generally intractable to calculate. We therefore need to approximate the expectation by sampling from the distributions, and calculating the Monte Carlo average. In practice, this can be done by gradient-based MCMC sampling method, such as Langevin Dynamics [64].

Given $\mathbf{x}$, we have $p_\theta(\mathbf{z}|\mathbf{x})\propto p_\beta(\mathbf{x}|\mathbf{z})p_\alpha(\mathbf{z})$. Note that

$$\begin{aligned}\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x}|\mathbf{z}) &= \frac{1}{p_\beta(\mathbf{x}|\mathbf{z})}\nabla_{\mathbf{z}}p_\beta(\mathbf{x}|\mathbf{z}) \\ &= \int_{\mathbf{w}} p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x},\mathbf{w}|\mathbf{z})d\mathbf{w} \\ &= \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z})}\left[\nabla_{\mathbf{z}}\log p_\beta(\mathbf{x},\mathbf{w}|\mathbf{z})\right].\end{aligned} \tag{18}$$

Therefore, the log-likelihood of surrogate target distribution for the Langevin dynamics at the t-th step is

$$\begin{aligned}\log\tilde{Q}(\mathbf{z}_t) &= \log p_\alpha(\mathbf{z}_t) + \mathbf{E}_{\mathbf{w}\sim p_\beta(\mathbf{w}|\mathbf{x},\mathbf{z}_t)}\left[\sum_{i=1}^{D}\sum_{k=1}^{2} w_{ik}\left(\log\pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right)\right] \\ &= \log p_\alpha(\mathbf{z}_t) + \sum_{i=1}^{D}\sum_{k=1}^{2}\gamma_{ik,t}\left(\log\pi_{ik} + \log p_{\beta_k}(\mathbf{x}_i|\mathbf{z}_{k,t})\right),\end{aligned} \tag{19}$$

which has the same form as $\mathcal{J}(\theta)$. However, instead of updating parameters θ, Langevin dynamics updates the latent variables $\mathbf{z}$ with the calculated gradients.

The two-step learning process of the DRC models can be understood as follows: (1) in the first step, the algorithm optimizes $\mathcal{J}$ by updating latent variables $\mathbf{z}$, where the posterior responsibility γ_{ik} inferred at each step serves to gradually disentangle the foreground and background components, and (2) in the second step, the updated $\mathbf{z}$ is fed again into the models to generate the observation $\mathbf{x}$, where the algorithm optimizes $\mathcal{J}$ by updating the model parameters θ.

It is worth mentioning that learning LEBMs requires an extra sampling step [47], as the gradients are given by the following

$$\delta_\alpha(\mathbf{x}) = \mathbf{E}_{p_\theta(\mathbf{z}|\mathbf{x})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right] - \mathbf{E}_{p_\alpha(\mathbf{z})}\left[\nabla_\alpha f_\alpha(\mathbf{z})\right], \tag{20}$$

where the second terms should be computed by sampling with $p_\alpha(\mathbf{z})$. We term this as *prior sampling* in the main paper.

Further Details on the Loss Functions For the generative models $p_{\beta_k}(\mathbf{x}|\mathbf{z}_k),\ k=1,2$, we assume that $\mathbf{x} = g_{\beta_k}(\mathbf{z}_k) + \epsilon$, where $g_{\beta_k}(\mathbf{z}_k),\ k=1,2$ are the generator networks for foreground and background regions, and ϵ is random noise sampled from a zero-mean Gaussian or Laplace distribution. Assuming a global fixed variance σ^2 for Gaussian, we have $\log p_{\beta_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{2\sigma^2}\left\|g_{\beta_k}(\mathbf{z}_k) - \mathbf{x}\right\|^2 + C,\ k=1,2$, where C is a constant unrelated to β_k and $\mathbf{z}_k$. Similarly for Laplace distribution, we have $\log p_{\beta_k}(\mathbf{x}|\mathbf{z}_k) = -\frac{1}{\lambda}|g_{\beta_k}(\mathbf{z}_k) - \mathbf{x}| + C,\ k=1,2$. These two log-likelihoods correspond to the MSE loss and L1 loss commonly used for image reconstruction, respectively.

C.2 Regularization

Pseudo Label Learning As mentioned in the paper, we exploit the symbolic vector $\mathbf{y}$ emitted by the LEBM for additional regularization. Let the target distribution of $\mathbf{y}_k$ be P_k given by $p_{\alpha_k}(\mathbf{y}|\mathbf{z}_k),\ k=1,2$, which represents the distribution of symbolic vector for foreground and background regions respectively. We can optimize the following objective as a regularization to our original learning objective:

$$\max_{\beta,\tau}\ \mathcal{L}_{\text{pseudo-label}} = \sum_{k=1}^{2} H(P_k, Q_k), \tag{21}$$

$$H(P_k, Q_k) = -\langle p_{\alpha_k}(\mathbf{y}|\mathbf{z}_k), \log q_{\tau_k}(\mathbf{y}|g_{\beta_k}(\mathbf{z}_k))\rangle,\ k = 1, 2, \tag{22}$$

where $q_{\tau_k},\ k = 1, 2$ represents the jointly trained auxiliary classifier network (see Appendix B for architecture details) for foreground and background. $g_{\beta_k}(\mathbf{z}_k),\ k = 1, 2$ represents the output of generator network. We set the weight of this regularization term to 0.1 for all the models.

Total Variation norm (TV-norm) Total Variation norm [75] is commonly used for image denoising, and has been extended as an effective technique for in-painting. We use TV-norm as a regularization for learning the background generator.

$$\min_{\beta_2}\ \mathcal{L}_{\text{TV-norm}} = \sum_{h,w}\left(|\frac{\partial g_{\beta_2}(\mathbf{z}_2)}{\partial x}(h, w)| + |\frac{\partial g_{\beta_2}(\mathbf{z}_2)}{\partial y}(h, w)|\right), \tag{23}$$

where $\partial_x g_{\beta_2}(\mathbf{z}_2)(h, w)$ and $\partial_y g_{\beta_2}(\mathbf{z}_2)(h, w)$ represent the horizontal and vertical image gradients at the pixel coordinate (h, w) respectively. We set the weight of this regularization term to 0.01 for all the models.

Orthogonal Regularization We use orthogonal regularization [74] for the convolutional layers only. Let $\mathbf{W}$ be the flattened kernel weights of the convolutional layers, *i.e.*, the size of $\mathbf{W}$ is $C \times K$ where C is the output channel number. The orthogonal regularization is calculated according to

$$\min_{\beta}\ \mathcal{L}_{\text{orthogonal-reg}} = \|\mathbf{W}\mathbf{W}^T \odot (\mathbf{1} - \mathbf{I})\|_F, \tag{24}$$

where $\odot$ is the Hadamard product. $\mathbf{I}$ denotes the identity matrix, and $\mathbf{1}$ denotes the matrix filled with ones. We set the weight of this regularization term to 0.1 for Birds models, and 1.0 for the rest of the models.

D Pytorch-style Code

We provide pytorch-style code to illustrate how the learning and inference in our model work.

Forward Pass In the forward pass, the model takes latent variables $\mathbf{z}$, generates foreground and background regions separately, and mixes them into the final image. Note that the pixel re-assignment is applied to both background image and mask. We finds it useful to feed the intermediate feature of background region into the generator for pixel re-assignment.

Listing 1: Forward pass of the DRC model.

```
def forward(z):
    zf, zb, zs = z[:, :ZF_DIM], \
                 z[:, ZF_DIM:-ZS_DIM] \
                 z[:, -ZS_DIM:]

    ### generating foreground
    fg, fm = fg_net(zf)

    ### generating background
    bg, bm, bg_feat = bg_net(zb)
    shuffling_grid = sp_net(zs, bg_feat.detach())
    bg_shuf = F.grid_sample(bg, shuffling_grid)
    bm_shuf = F.grid_sample(bm, shuffling_grid)

    ### generating foreground masks
    pi = torch.cat([fm_wp, bm_wp], dim=1).softmax(dim=1)
    pi_f, pi_b = pi[:,:1,...], pi[:,1:,...]

    ### mixing regions
    im_p = fg * pi_f + bg_wp * pi_b
    return im_p, fg, bg_shuf, pi_f, pi_b, bg
```

Sampling Latent Variables We employ Langevin Dynamics for sampling latent variables, which iteratively updates the sample with the gradient computed against the likelihood. In the following code, `ebm_net` stands for the LEBMs, which outputs the energy and the distribution paramters of the symbolic vector $\mathbf{y}$ for the foreground and background regions.

Listing 2: Running Langevin Dynamics for prior and posterior sampling of the latent variables $\mathbf{z}$.

```
def sample_langevin_prior(z):
    ### langevin prior inference
    # only latent variables 'z' are updated
    for __ in range(infer_step_K0):
        z = z.requires_grad_(True)

        en, __, __ = ebm_net(z)
        e_log_lkhd = en.sum() + .5 * z.square().sum()

        d_ebm_z = torch.autograd.grad(e_log_lkhd, z)[0]
        z = z - 0.5 * (self.delta_0 ** 2) * d_ebm_z \
                    + self.delta_0 * torch.randn_like(z)
    return z

def sample_langevin_posterior(z, im_t):
    ### langevin posterior inference
    # only latent variables 'z' are updated
    for __ in range(infer_step):
        z = z.requires_grad_(True)

        im_p, fg, bg_shuf, pi_f, pi_b, bg = forward(z)

        ### log-lkhd for LEBMs
        en, __, __ = ebm_net(z)

        ### log-lkhd for generators
        log_pf = - F.l1_loss(fg, im_t, reduction='none')
                 / (2. * SIGMA ** 2)
        log_pb = - F.l1_loss(bg_shuf, im_t, reduction='none')
                 / (2. * SIGMA ** 2)

        # posterior responsibility
        with torch.no_grad():
            ga_f = pi_f * log_pf.exp() /
                   (pi_f * log_pf.exp()
                  + pi_b * log_pb.exp() + 1e-8)
        # objective for image generation
        e_z_log_p = ga_f.detach() * \
                    ((pi_f + 1e-8).log() + log_pf) \
                  + (1. - ga_f.detach()) * \
                    ((pi_b + 1e-8).log() + log_pb)
        # regularization
        tv_norm = tv_loss(bg_shuf)

        j_log_lkhd = - e_z_log_p.sum() + tv_norm * .01 + \
                     + en.sum() + .5 * z.square().sum()

        d_j_z = torch.autograd.grad(j_log_lkhd, z)[0]
        z = z - 0.5 * (self.delta_1 ** 2) * d_j_z \
              + self.delta_1 * torch.randn_like(z)
        z = z.detach()
    return z
```

Updating Model Parameters Given the sampled latent variables $\mathbf{z}$, we optimize the model parameters by minimizing the reconstruction error.

Listing 3: Updating model parameters.

```
def update_G(im_t, fg, bg_shuf, pi_f, pi_b, bg,
             zf_logits, zb_logits):
    ### optimizers for generator networks
    fg_net_optimizer.zero_grad()
    bg_net_optimizer.zero_grad()
    sp_net_optimizer.zero_grad()
    ### optimizers for auxiliary classifiers
    fc_net_optimizer.zero_grad()
    bc_net_optimizer.zero_grad()

    ### Regularizations
    # Pseudo-label for additional regularization
    f_logits = fc_net(fg)
    b_logits = bc_net(bg)
    hpq_f = cross_ent(zf_logits, f_logits)
    hpq_b = cross_ent(zb_logits, b_logits)
    # orthogonal regularizations
    ortho_reg = orthogonal_reg(fg_net) + \
                    orthogonal_reg(bg_net)
    # TV-norm
    tv_norm = tv_loss(bg_shuf)

    ### log-lkhd for generators
    log_pf = - F.l1_loss(fg, im_t, reduction='none')
             / (2. * SIGMA ** 2)
    log_pb = - F.l1_loss(bg_shuf, im_t, reduction='none')
             / (2. * SIGMA ** 2)

    # posterior responsibility
    with torch.no_grad():
        ga_f = pi_f * log_pf.exp()
             / (pi_f * log_pf.exp() + pi_b * log_pb.exp() + 1e-8)
    # objective for image generation
    e_z_log_p = ga_f.detach() * ((pi_f + 1e-8).log() + log_pf) \
       + (1. - ga_f.detach()) * ((pi_b + 1e-8).log() + log_pb)

    G_loss = - e_z_log_p.mean() + tv_norm * .01 + \
             hpq_f * .1 + hpq_b * .1 + ortho_reg * 1

    G_loss.backward()
    fg_net_optimizer.step()
    bg_net_optimizer.step()
    sp_net_optimizer.step()

    fc_net_optimizer.step()
    bc_net_optimizer.step()

def update_E(en_pos, en_neg):
    ebm_optimizer.zero_grad()

    ebm_loss = en_pos.mean() - en_neg.mean()
    ebm_loss.backward()

    ebm_optimizer.step()

def learn(zp, zn, im_t):
    ### 1. Sampling latent variables
    zp = sample_langevin_posterior(zp)
    zn = sample_langevin_prior(zn)

    ### 2. Updating the parameters
```

```
en_pos, zpf_logits, zpb_logits = ebm_net(zp)
en_neg, znf_logits, znb_logits = ebm_net(zn)
# update LEBMs
update_E(en_pos, en_neg)

im_p, fg, bg_shuf, pi_f, pi_b, bg = forward(zp)
# update the generators
update_G(im_t, fg, bg_shuf, pi_f, pi_b, bg,
         zf_logits, zb_logits)
```

E Evaluation Protocols

Intersecion of Union (IoU) The IoU score measures the overlap of two regions A and B by calculating the ratio of intersection over union, according to

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{25}$$

where we use the inferred mask and ground-truth mask as A and B respectively for evaluation.

Dice (F1) score Similarly, the Dice (F1) score is

$$\text{Dice}(A, B) = \frac{2|A \cap B|}{|A| + |B|}. \tag{26}$$

Higher is better for both scores.

Evalution As mentioned in the paper, IODINE [37] and Slot-attention [38] are designed for segmenting complex multi-object scenes using slot-based object representations. Ideally, the output of these models consists of masks for each individual object, while the background is viewed as a virtual "object" as well. In practice, however, it is possible that the model distributes the background over all the slots as mentioned in Locatello et al. [38]. Taking both cases into consideration (see Fig. S3 and Fig. S4), we propose two approaches to convert the multiple output masks into a foreground-background partition, and report the best results of these two options: (1) we compute the scores by making each mask as the background mask at a time, and then choose the best one; this works better when the background is treated as a virtual "object"; (2) we threshold and combine all the masks into a foreground mask; this is for when background is distributed to all slots.

Figure S3: An example situation when using each individual mask as the background mask gives higher scores. Note that if we threshold the output of each individual slot and compose them, the result would be the mask shown in the last column.

Figure S4: An example situation when thresholding&combining the output of each individual slot gives higher scores. We can see from the last column that the combined mask fits the foreground objects well.

F Additional Illustrations and Baseline Results

F.1 More Examples

We provide more foreground extraction results of our model for each dataset; see Fig. S5, Fig. S6, Fig. S7 and Fig. S8. From top to bottom, we display: (i) observed images, (ii) generated images, (iii) masked generated foregrounds, (iv) generated backgrounds, (v) ground-truth foreground masks, and (vi) inferred foreground masks in each figure.

F.2 Failure Modes

We provide examples for illustrating typical failure modes of the proposed model; see Fig. S9. On Birds dataset, we observe that the method can perform worse on samples where the foreground object has colors and textures quite similar to the background regions. Although the method can still capture the rough shape of the foreground object, some details can be missing. On TM-dSprites dataset, we observe that the method may occasionally miss one of the foreground objects. We conjecture that the problem can be mitigated with more powerful generator and further fine-tuning on this dataset.

F.3 Baseline Results

GrabCut We provide results of GrabCut [5] on Birds dataset and TM-dSprites dataset, shown in Fig. S10. We can see that GrabCut algorithm may fail when the foreground object and background region have moderately similar colors and textures. On TM-dSprites dataset, GrabCut algorithm outperforms other baselines, but is still inferior to the proposed method and exhibits a similar failure pattern.

ReDO We provide results of ReDO [28] on Birds dataset and TM-dSprites dataset, shown in Fig. S11. ReDO overall performs better than GrabCut on Birds dataset, while it may fail when the background regions become more complex. We can also observe that ReDO relies heavily on the pixel intensities for foreground-background grouping on TM-dSprites dataset.

IODINE On Birds dataset, we observe that IODINE [37] tends to use color as a strong cue for segmentation, see Fig. S12. On TM-dSprites dataset, IODINE is distracted by the background; see Fig. S13. These two findings are consistent with those reported in [37];

Slot-Attention On Birds dataset, Slot-attention learns to roughly locate the position of foreground objects, but mostly fails to provide foreground masks when the background region becomes complex; see Fig. S14. Similarly, we can observe that Slot-Attention tends to use color as a strong cue for segmentation. On TM-dSprites dataset, Slot-attention is distracted by the background; see Fig. S15.

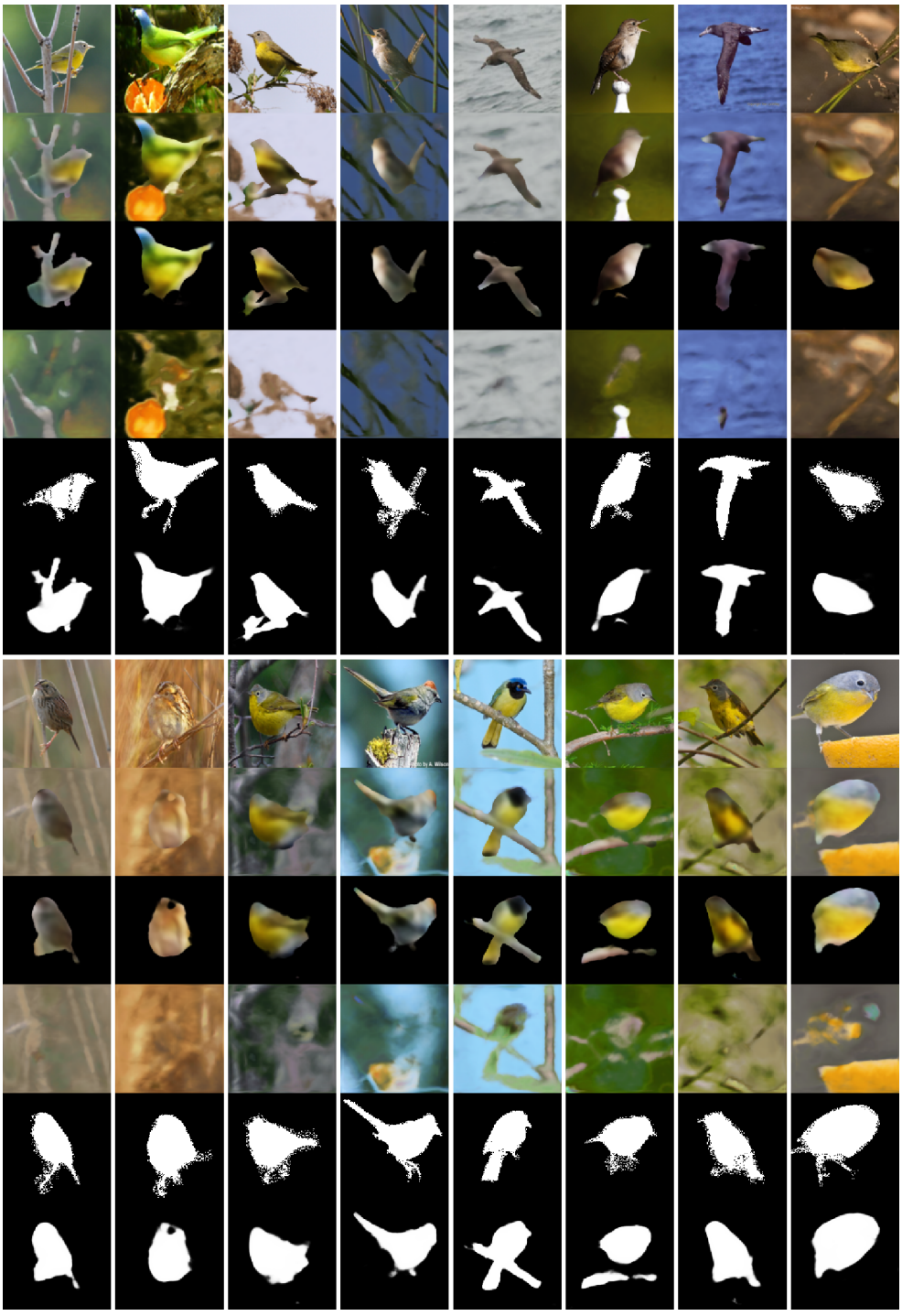

Figure S5: Additional foreground extraction results on Birds dataset.

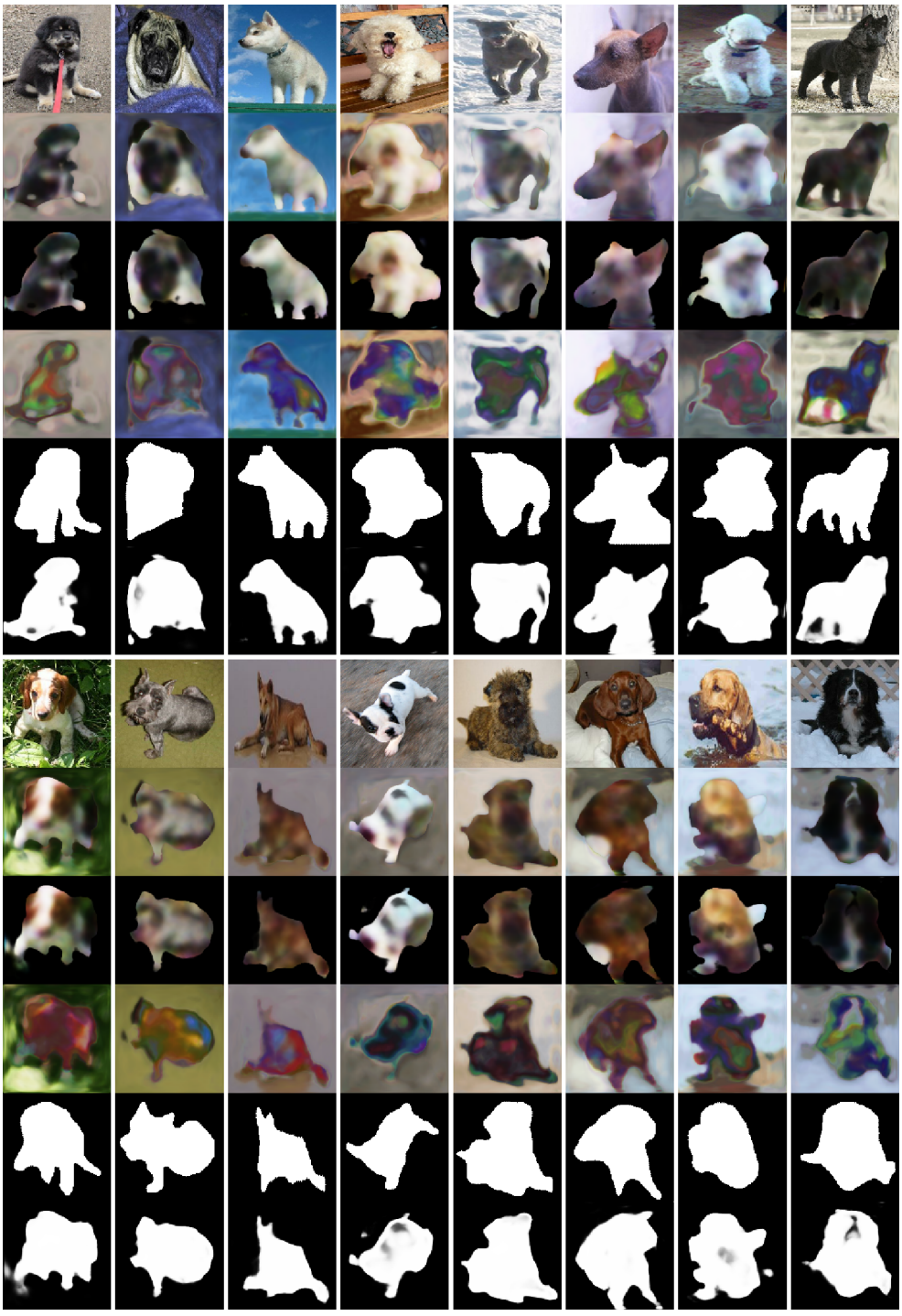

Figure S6: Additional foreground extraction results on Dogs dataset.

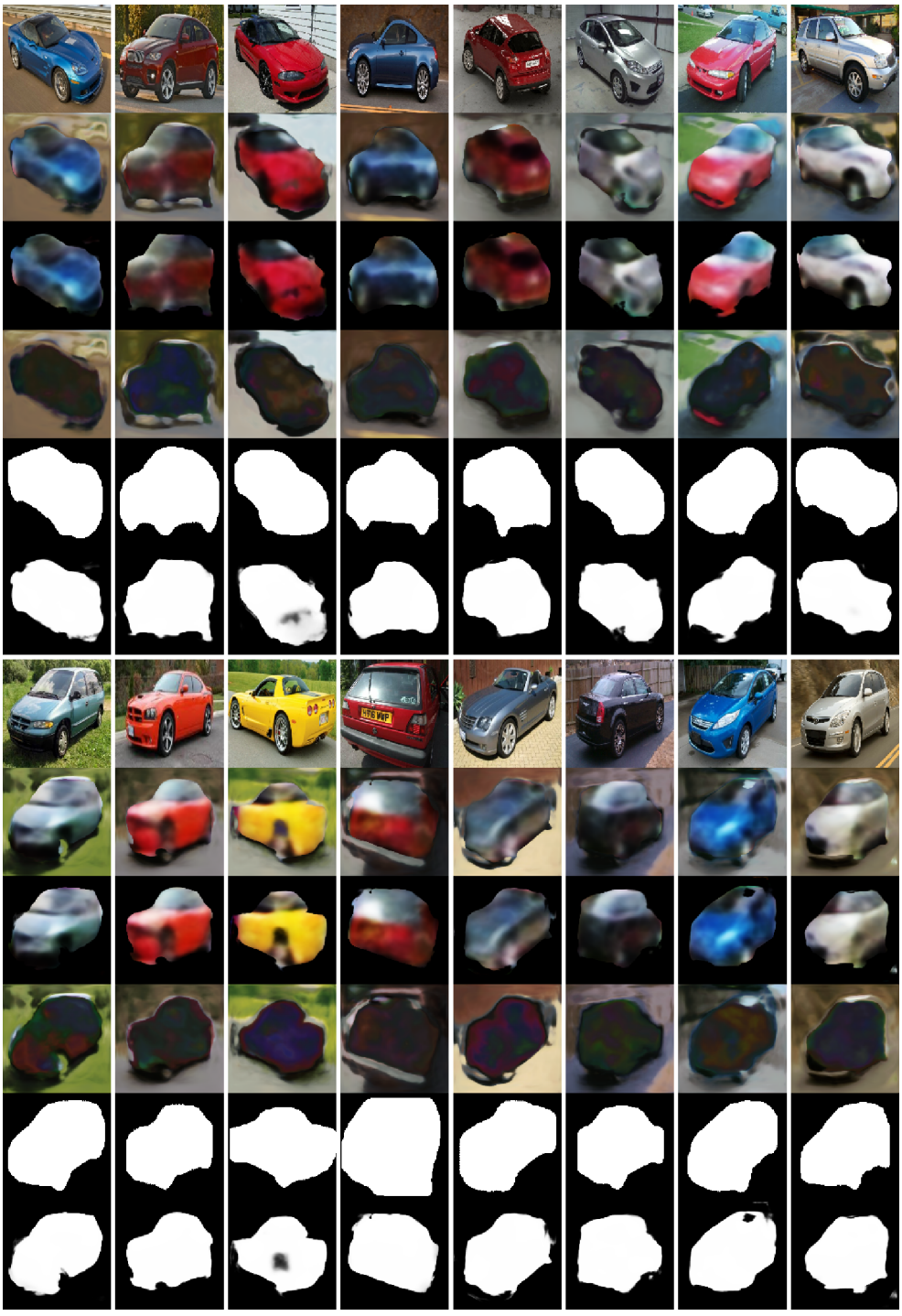

Figure S7: Additional foreground extraction results on Cars dataset.

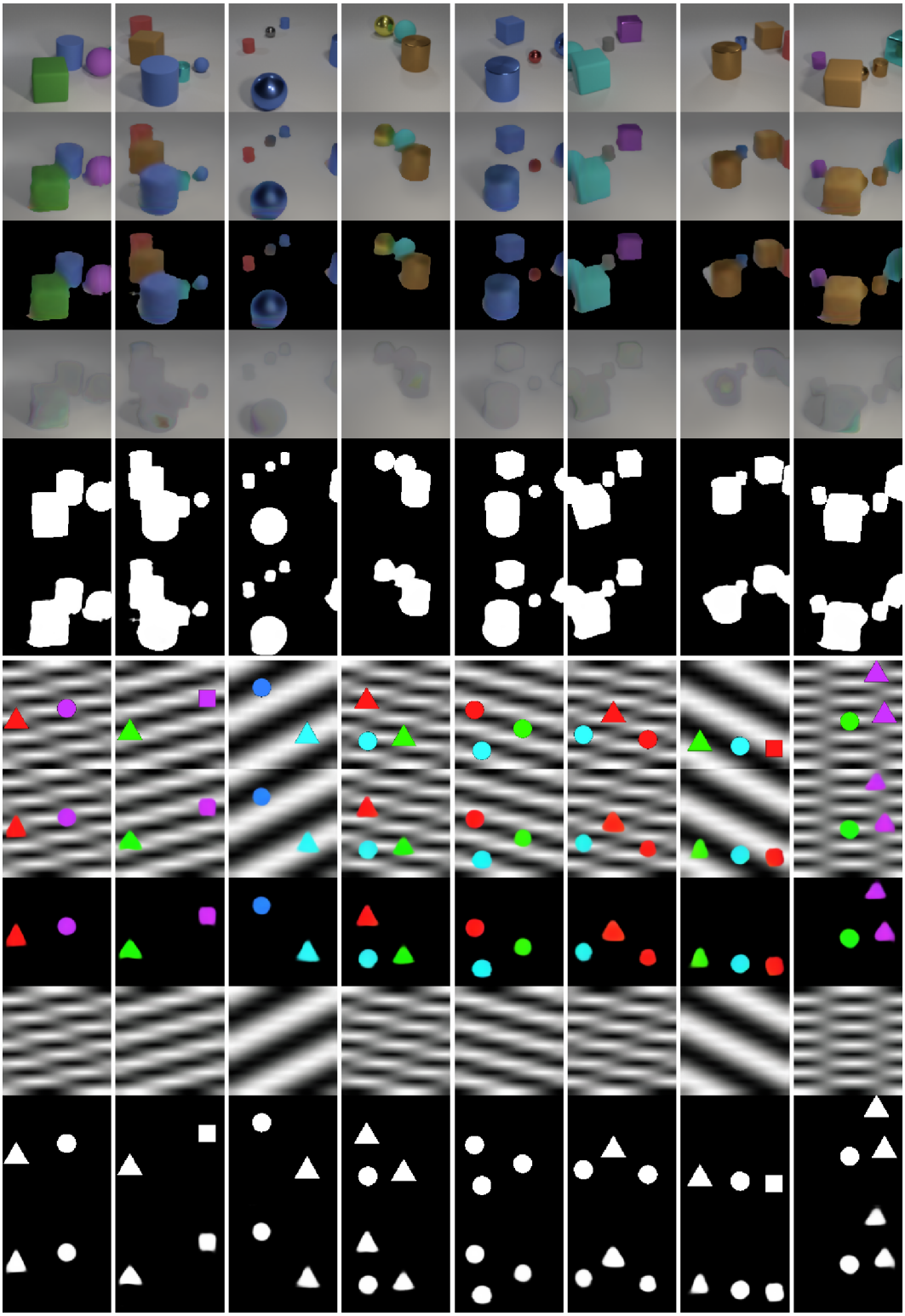

Figure S8: Additional foreground extraction results on CLEVR6 and TM-dSprites datasets.

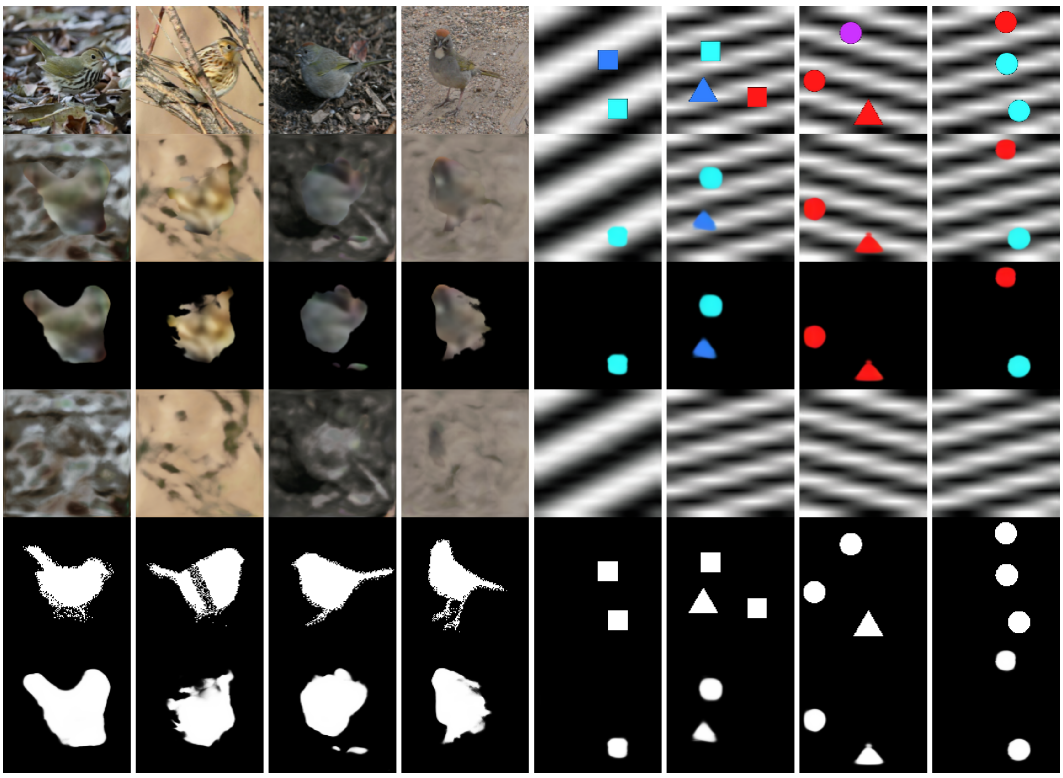

Figure S9: Typical failure modes on Birds and TM-dSprites.

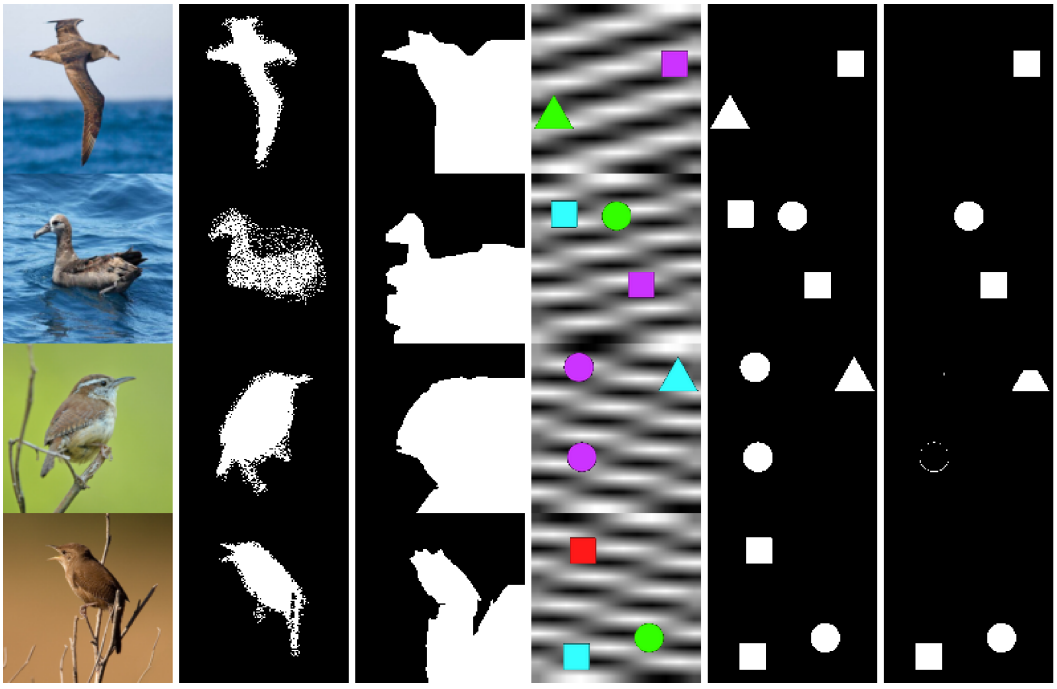

Figure S10: Results of GrabCut on Birds and TM-dSprites datasets. The first three columns are results from Birds dataset, and the last three are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, and the foreground extraction results respectively.

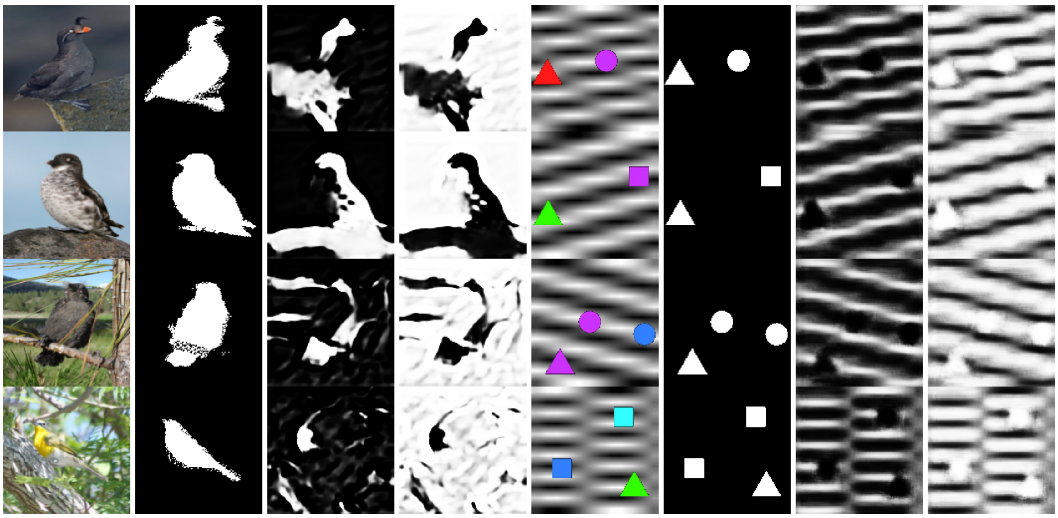

Figure S11: Results of ReDO on Birds and TM-dSprites datasets. The first four columns are results from Birds dataset, and the last four are from TM-dSprites. From left to right, we display the observed image, ground-truth mask, mask from the first output channel and from the second channel respectively.

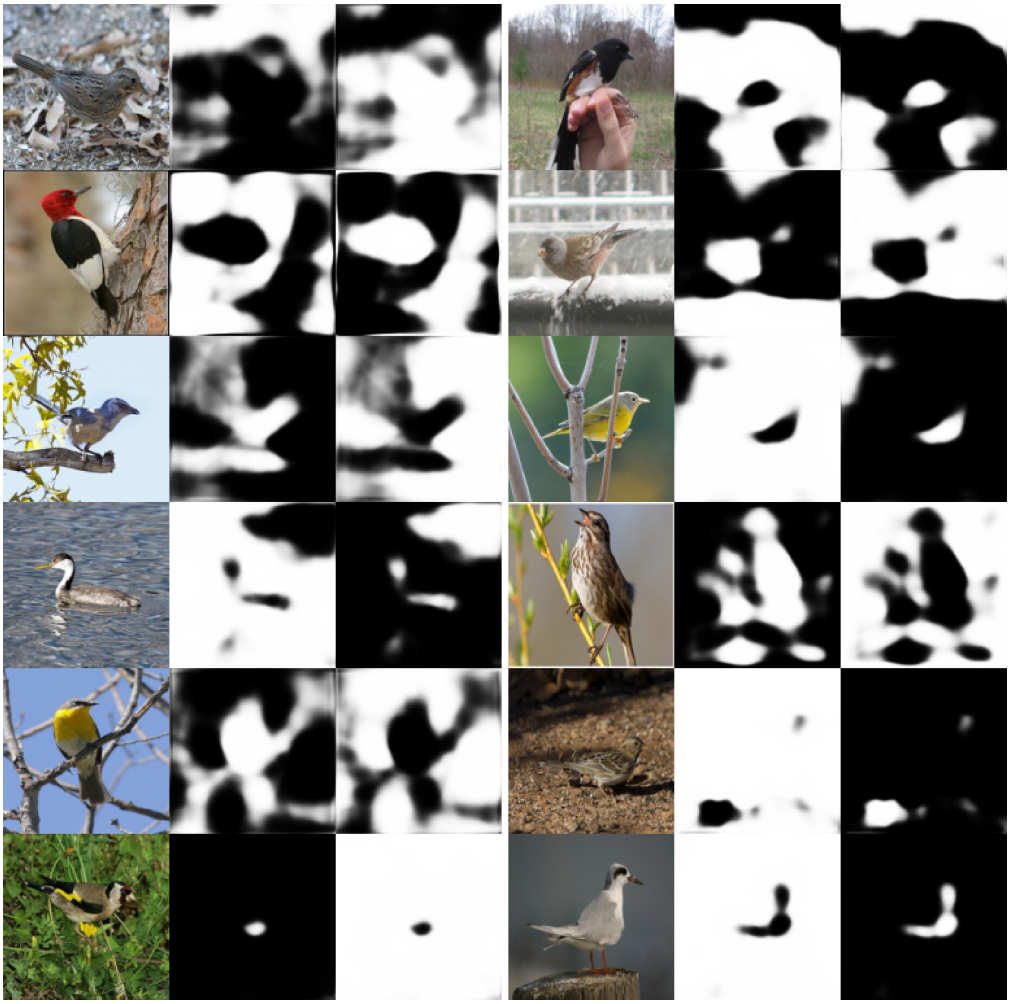

Figure S12: Results of IODINE on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

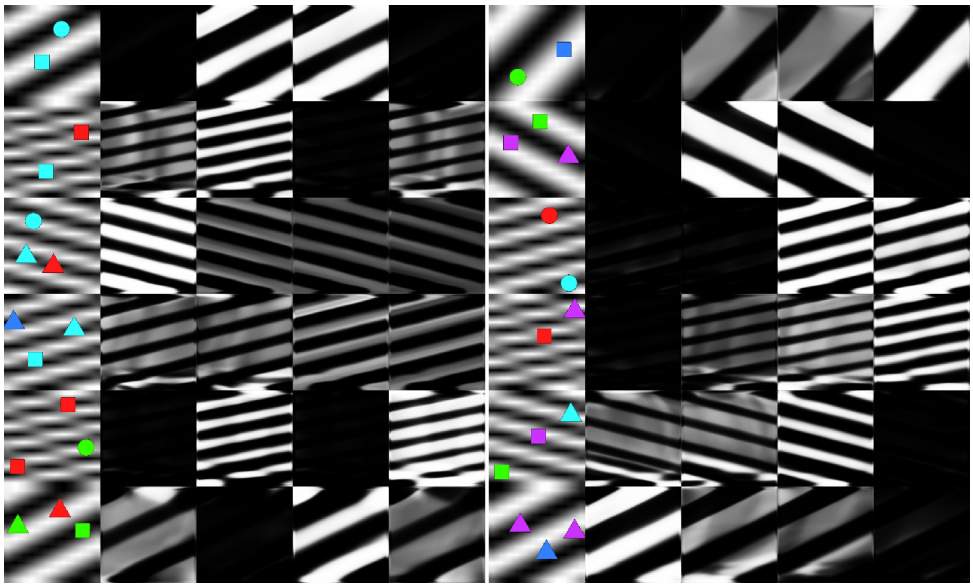

Figure S13: Results of IODINE on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

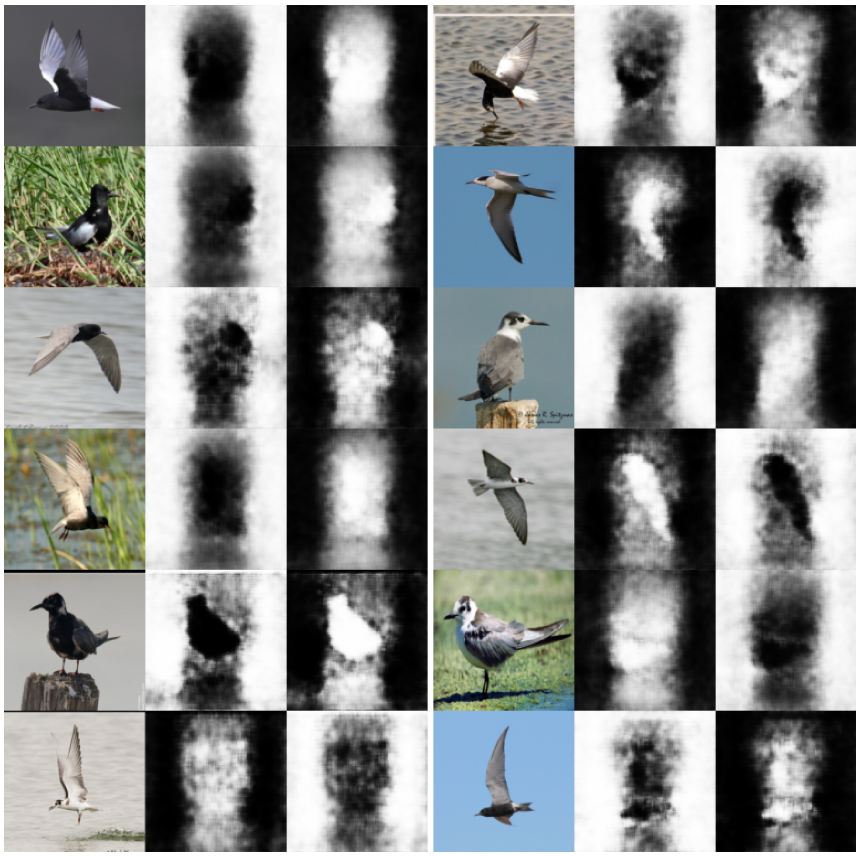

Figure S14: Results of Slot-Attention on Birds datasets. We provide the observed image, mask from the first slot and from the second slot respectively.

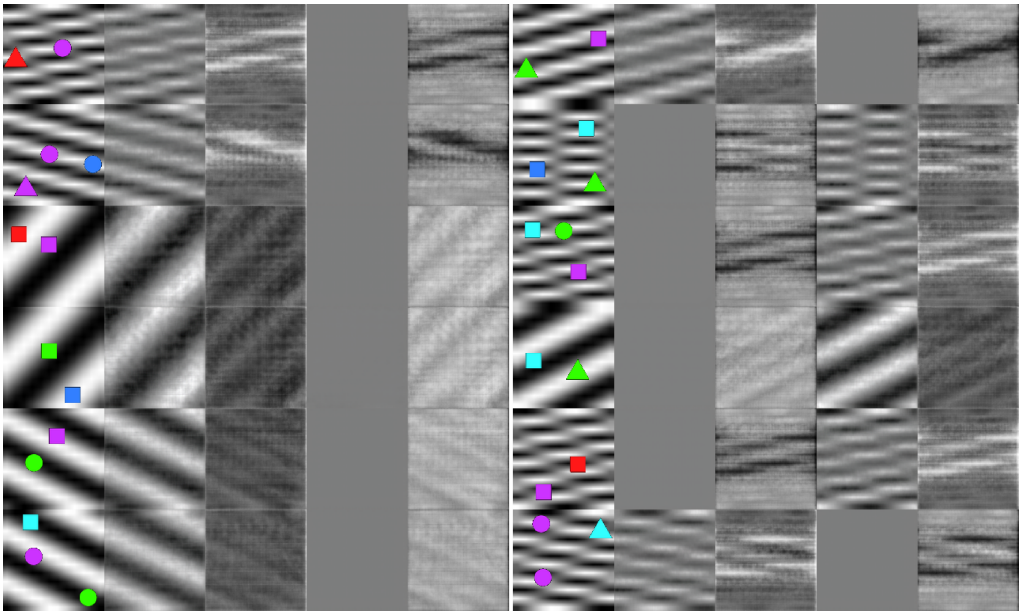

Figure S15: Results of Slot-Attention on TM-dSprites datasets. We provide the observed image and masks from four object slots respectively.

G Further Discussion

G.1 Preliminary Analysis of Real-world Datasets

We provide preliminary analysis of the statistics of the three real-world datasets. To measure the similarity of colors and textures for these datasets, we calculate the image histogram for the foreground objects and background regions of each dataset; see Fig. S16. To probe the similarity of shape distributions, we also provide the heatmap of foreground masks, as shown in Fig. S17. The heatmaps are calculated by overlapping the ground-truth masks and normalizing the summarized intensities with the maximum values. Despite the apparent difference in Birds vs Dogs and Cars, we can see that the data distribution of Birds dataset is more similar to that of Dogs dataset than to that of Cars dataset. We can also observe the similarity between the distributions of Dogs and Cars datasets. This could partly explain why the proposed method shows relatively strong performance on objects from unseen categories, *i.e.*, it effectively combines the colors, textures and shapes for foreground extraction.

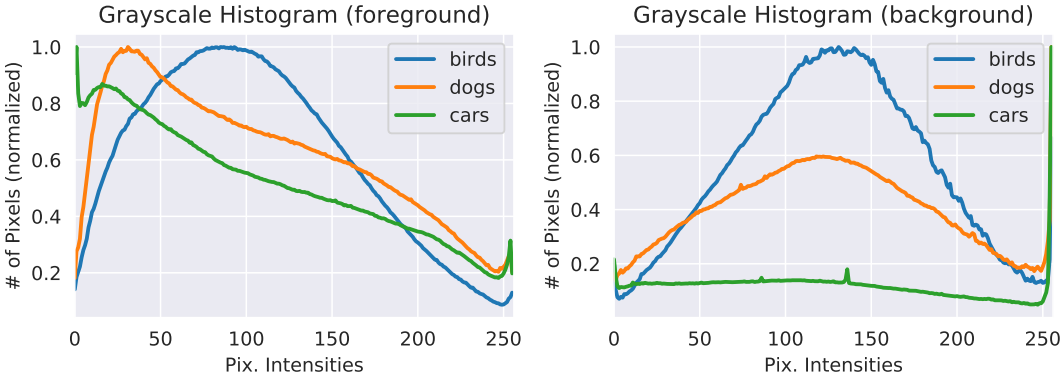

Figure S16: Image histograms for foreground objects and background regions from each dataset.

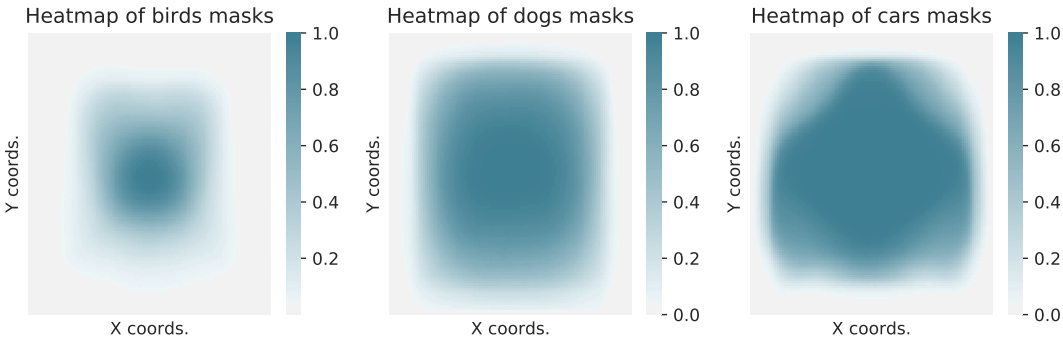

Figure S17: Heatmaps of ground-truth masks for each dataset.

G.2 Possible Extension to Multi-Object Segmentation

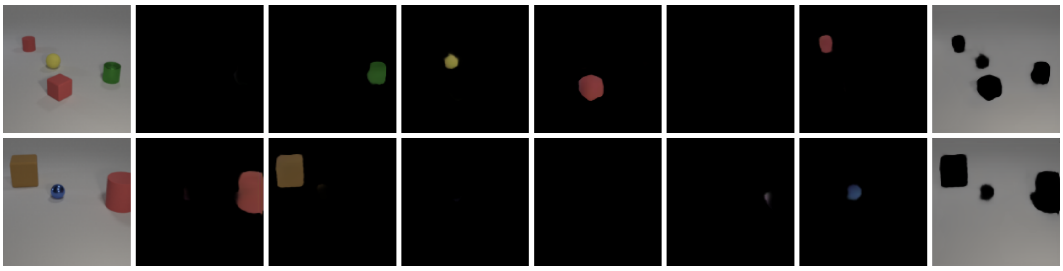

Figure S18: Preliminary results on learning slot-based object representation.

We explore the possibility of using our model for segmenting and disentangling multiple objects. As shown in Fig. S18, the proposed method can disentangle the foreground objects, while providing explicit identification of the background region. However, we find that the model occasionally distributes a single object into several slots based on the difference in texture and shading; see Fig. S19. We conjecture that this is due to the lack of objectness modeling. We would like to investigate more on this direction in future works.

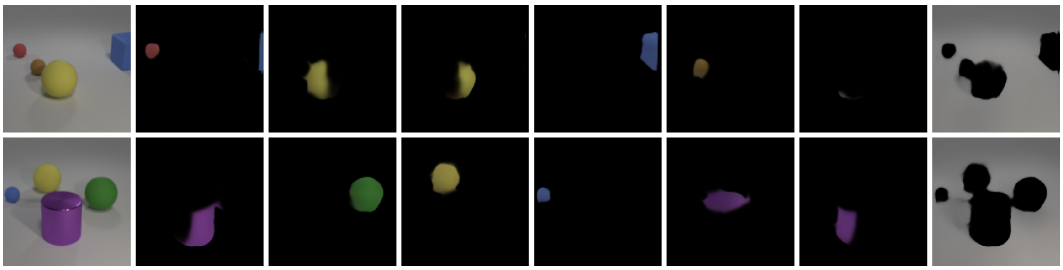

Figure S19: Failure modes of energy-based slot representation model.

Figure S20: Prior sampling results on Birds dataset.

G.3 Prior Sampling Results on Birds Dataset

We provide preliminary results of sampling from the learned energy-based priors, as shown in Fig. S20. Of note, the generated prior samples are generally less realistic compared with the posterior samples, as prior sampling does not involve the region competition between foreground and background components, which may lead to worse separation and the generation of foreground and background regions. We would further explore generating foreground and background in future work.