DENF: UNSUPERVISED SCENE-DECOMPOSITIONAL NORMALIZING FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised object-centric scene decomposition models can learn compositional and hierarchical representations of multi-object scenes that allow abstraction into object entities and spaces. However, previous approaches, either based on VAE or GAN frameworks, have no guarantee of preserving particular aspects of the image in scene reconstruction. In this work, we propose the first probabilistic model based on normalizing flows. The scene is represented as mixture of bidirectional flows that map a set of structured prior distributions into the scene data distribution. The bijective mapping of our algorithm (DeNF) yields an efficient sampling and density evaluation in training time. Furthermore, it improves the fidelity of the scene's visual contents in the reconstruction process. In our experiments on real and synthetic image data for unsupervised scene decomposition, DeNF achieves competitive results.

1 INTRODUCTION

This paper is about learning disentangled representations from images depicting one or more foreground objects without using any labeled data. This problem is known as unsupervised object-centric scene representation. We consider two tasks, namely to both decompose a given image and generate a new image from a structured latent representation by bidirectional normalizing flows. The structured latent space may include the disentangled geometric representation of objects such as position and scale and the photometric representation such as color and texture. Structuring the latent variables in the learning model introduces an inductive bias to decompose the input scene into object instances and spaces. This explicit representation of objects, in contrast to the implicit representation of deep learning methods with unstructured latent spaces, is beneficial as it can generalize better to novel compositions Greff et al. (2020).

Although unsupervised learning of object-centric scene decomposition continues to be a challenging task, there has been impressive progress in the past few years with promising results on multi-object synthetic datasets. These methods are primarily based on generative adversarial networks (GAN) Goodfellow et al. (2014) or deep variational auto-encoders (VAE) Kingma & Welling (2014).

In GAN-based object-centric generative models Phuoc et al. (2020); Ehrhardt et al. (2020); Kodali et al. (2017); Radford et al. (2016); Niemeyer & Geiger (2021); Stelzner et al. (2021), the generator network receives structured noise and transforms it into images by minimizing a discrepancy loss between the real examples and the generated ones. Due to their adversarial setting, these models may produce complex visual scenes with cluttered backgrounds. However, they are not capable of decomposing a scene. Moreover, these models are computationally expensive and hard to train.

VAE-based object-centric generative models Burgess et al. (2019); Greff et al. (2019); Engelcke et al. (2019); Crawford & Pineau (2019); Lin et al. (2020); Zhu et al. (2021) can decompose the scene by structuring the latent variables at the output of an encoder network. By imposing a set of prior distributions on the latent variables and maximizing the evidence lower bound (ELBO), the representations of objects and background are encoded into the latent variables used by the decoder network to reconstruct the scene. While VAE-based models have been shown to attain proper decompositions, they provide no guarantee in preserving any particular aspect of the scene. This is mainly due to decomposing and generating the image through two separate encoder and decoder modules. Although minimizing the reconstruction loss forces the decoder to learn the reverse

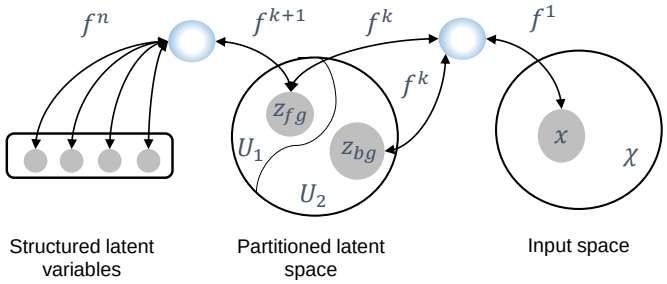

Figure 1: A mixture of flows (i.e., bijective mappings f^i) generates the scene image (x) from a set of latent variables. The latent space of the model is partitioned into the foreground (z_{fg}) and background (z_{bg}) components that generate non-overlapping regions of the image. The foreground component is further decomposed into a set of structured latent variables that represent the objects.

mapping of the encoder on the training data, any unseen variations to the scene may be considered out of distribution and might not be preserved in the reconstruction.

In contrast to existing frameworks, we establish our work on flow-based generative models Rezende & Mohamed (2015); Dinh et al. (2014; 2016); Papamakarios et al. (2021). We present a scene-decompositional normalizing flow (DeNF), an unsupervised probabilistic model that formulates the distribution of scene data with a mixture of prior distributions on a set of structured latent variables (see Figure 1). To the best of our knowledge, this is the first approach in which normalizing flows are decomposed into structured latent variables. The invertible architecture of DeNF makes it possible to perform both sampling and density evaluation in an efficient and exact way for decomposing and reconstructing the scene. For training, two forward and backward processes are involved. By mapping the given scene images into a set of *base* (i.e., prior) distributions which represent the background and foreground components, density estimation is used for updating the parameters of mapping. Reversely, the scene is reconstructed and compared with the original data by constraining and sampling from the base distributions and inverting the mapping. Once trained, DeNF is capable of reconstructing a diverse set of scene data by preserving only a subset of objects without losing information pertaining to the background. Such fidelity to original scene reconstruction is inherently a result of the bijective mapping of DeNF Liu et al. (2020).

2 Related Work

Several methods have been proposed to perform unsupervised object-oriented image (de)composition in the last few years. These methods can be roughly categorized into (1) GAN-based frameworks and (2) VAE-based models.

GAN-based object-oriented scene generation approaches Radford et al. (2016); van Steenkiste et al. (2020); Phuoc et al. (2020); Ehrhardt et al. (2020); Niemeyer & Geiger (2021); Stelzner et al. (2021) mostly generate the visual scene from a set of structured latent variables (i.e., noise) that introduce interpretable representations for the objects in the scene. Depending on the architecture, the generator can be implemented by a convolutional network or by a Neural Radiance Fields (NeRFs) renderer Mildenhall et al. (2020). Often, it is trained along with a discriminator network by playing a minimax game. Although these approaches can generate complex scenes with multiple objects and cluttered backgrounds, they are not capable of decomposing an image. Moreover, GAN-based approaches are computationally expensive and have stability issues during optimization.

Recent object-centric scene decomposition models based on VAE frameworks have shown promising results Burgess et al. (2019); Greff et al. (2019); Engelcke et al. (2019); Crawford & Pineau (2019); Lin et al. (2020); Zhu et al. (2021). MONet Burgess et al. (2019) relies on training a VAE together with a deterministic recurrent attention network to provide attention masks around image regions. However, the recurrent attention causes a propensity for scalability issues Engelcke et al. (2019). IODINE Greff et al. (2019) models the scene as a spatial mixture of objects and the background, then uses amortized iterative refinement of latent object representations within the variational framework,

which is expensive both in terms of computation and memory, limiting its practicality and utility. GENESIS Engelcke et al. (2019) also employs a spatial mixture model, which is encoded by component-wise latent variables. It parameterizes a spatial GMM over images decoded from a set of object-centric latent variables that are either inferred sequentially in an amortized fashion or sampled from an autoregressive prior. In GENESIS-V2 Engelcke et al. (2021), a modified expectation-maximization is used to infer the latent structure.

Other VAE-based models utilize a spatial attention mechanism for learning the latent structure that represents objects with *what*, *where*, and *presence* semantics. The AIR Eslami et al. (2016) uses an RNN to process the objects in the scene sequentially. This process does not scale well to images with many objects. SPAIR Crawford & Pineau (2019) addresses this shortcoming by replacing the RNN with a CNN. At the core of SPAIR is a convolutional object detector, similar to that used in YOLO Redmon et al. (2016), which maps from image to objects. This allows the model to specify the locations of objects relative to local grid cells rather than the entire image that facilitates spatially invariant computations. Based on this, the encoder maps the images into the feature volume with a predefined grid. Then, all the cells in the grid are processed sequentially to produce the objects in the scene. Similar to SPAIR, SPACE Lin et al. (2020) detects foreground objects on a grid, but the process is performed in parallel, thus it scales well to a larger number of objects. In GMAIR Zhu et al. (2021), a spatial attention mechanism combined with a Gaussian mixture prior clusters the input into discovered objects in a VAE-based model. This allows the model to categorize the *what* latent variables of objects additionally.

Structuring the latent variables in DeNF was inspired by prior attention-based works Eslami et al. (2016); Crawford & Pineau (2019); Lin et al. (2020). The objects are represented in the scene by *what*, *where*, and their *presence* latent variables. That being said, DeNF is based on a normalizing flows framework quite different from previous works.

Our contributions are as follows.

- We introduce DeNF, a novel unsupervised scene decompositional model, by formulating the probability distribution of images as a mixture of base distributions over a set of structured latent variables. The bijective mapping of the neural architecture enables both density estimation and sampling to be jointly used for training. In contrast to VAE-based methods, DeNF has high fidelity to the input image contents by preserving the visual information due to its bijective mapping.
- Inspired by prior attention-based models, DeNF employs a spatial transformer that is scalable to a large number of objects in the scene. Unlike previous works, the spatial transformer is used for disentangling the structured latent variables.

In the following, we describe the proposed DeNF model in detail.

3 Decompositional Normalizing Flows

The main idea of DeNF, illustrated in Figure 2, is to decompose and then reconstruct the scene with a mixture of objects and background latent variables. Each object is represented with a set of structured latent variables through bijective mappings in its architecture. The model learns a mapping between the probability distribution of the scene images and a set of base probability distributions (e.g., Gaussian distributions). To describe the probabilistic modeling of the DeNF, we briefly explain the mechanism of normalizing flows next.

3.1 Probabilistic Scene Modeling

3.1.1 Normalizing Flows

Normalizing flows (NFs) Dinh et al. (2014); Rezende & Mohamed (2015) provide a general way of constructing flexible probability distributions over continuous random variables. Let $x \in \mathbb{R}^d$ be a real vector, and suppose we would like to define a distribution $p(x)$ over x. The main idea of flow-based modeling is to express x through a learned invertible transformation $x = f_\theta(z)$ that warps a real vector z sampled from a *base* distribution $p_z(z)$. Making use of the change of variables, $p(x)$ can be

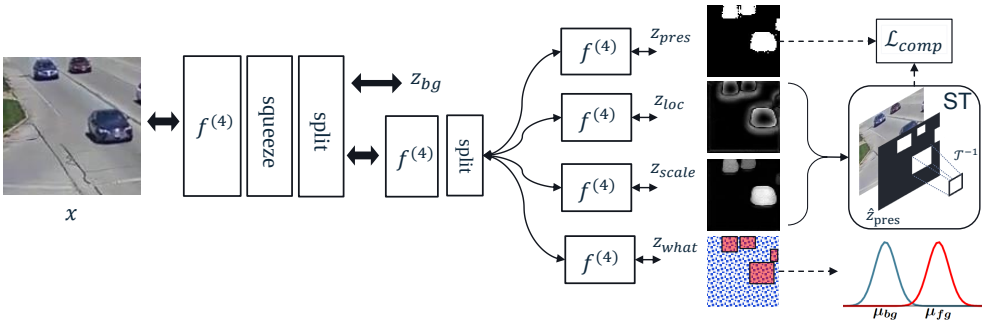

Figure 2: DeNF (de)composes the scene with a mixture of flows by stacking bijective mappings ($f^{(k)} = f^1 \circ ... \circ f^k$). The probability distribution of input $p(x)$ is partitioned into background ($p(z_{bg})$) and foreground $p(z_{fg})$ probabilities. The foreground components are a mixture of structure latent variables characterizing the objects in the scene. A spatial transformer (ST) is used to disentangle the representations learned by the structured variables.

written:

$$\log p_\theta(x) = \log p_z(z) + \log \left| det \left(\frac{\partial f_\theta^{-1}(x)}{\partial x} \right) \right|. \tag{1}$$

Typically in flow-based models, $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ is constructed by stacking several invertible and differentiable mappings, i.e., $f_\theta = f_\theta^1 \circ ... \circ f_\theta^L$ and obtain the log-determinant as the sum of the log-determinants of all the mappings. Equation 1 allows for exact maximum likelihood learning, which is not tractable in VAE and GAN-based generative models.

DeNF represents the measured data from a scene as the composition of two independent generative processes that construct the background and foreground parts of the scene. We extend the mapping f_θ to a mapping of background (z_{bg}) and foreground (z_{fg}) components to the non-overlapping elements of $x \in \{x_i\}_{i \in I}$ e.g. pixels of an image. This *many-to-one* transformation Papamakarios et al. (2021) can be specified as:

$$\begin{aligned} p_x(x) &= \sum_{i \in I} p_z(f_i^{-1}(x)) \left| det J_{f_i^{-1}}(x) \right| \\ &= p_{z_{fg}}(f_{fg}^{-1}(x)) \left| det J_{f_{fg}^{-1}}(x) \right| + p_{z_{bg}}(f_{bg}^{-1}(x)) \left| det J_{f_{bg}^{-1}}(x) \right|, \end{aligned} \tag{2}$$

where $J_f(.)$ is Jacobian of map f. The base distributions $p_{z_{bg}}$ and $p_{z_{fg}}$ correspond to background and foreground, respectively, and are explained in sections 3.1.2 and 3.1.3.

DeNF has a hierarchical architecture in mixing the latent variables of scene background with the structured latent variables that represent objects (see Figure 2). Inspired by the *multi-scale* architecture in Dinh et al. (2016), DeNF has *squeeze* and *split* layers for implementing a hierarchical flow-based neural architecture. An invertible flow-based model requires the dimensionality of z and x to be identical. To this end, a multi-scale transformation is used to reduce the computational cost and memory requirement of implementing f_θ by L layers. It clamps the number of subdimensions of $z_{k<L}$ and leaves them out for applying further transformations. This also provides a natural choice for granular data types such as images and waveforms Papamakarios et al. (2021); Dinh et al. (2016). Building upon this concept, the background representation is branched off earlier in the DeNF flows while the foreground flow undergoes further transformations. Coupling layers Dinh et al. (2016) are used for the invertible blocks in DeNF. A detailed explanation of the proposed architecture can be found in the Appendix.

3.1.2 Background Generative Flow

In the hierarchical DeNF architecture, the input image ($x \in \mathbb{R}^d$) is first mapped through four invertible blocks to a latent space ($z \in \mathbb{R}^d$) and is then split into two partitions. The first part is identified as background component $z_{bg} \in \mathbb{R}^{<d}$ while the second part is further processed through deeper layers of the network. The variables z_{bg} that are factored out at a finer scale (in an earlier layer)

are Gaussianized $z_{bg} \sim N(0, 1)$. The flow-based formulation allows for exact maximum likelihood learning. The forward Kullback-Leibler (KL) divergence between flow-based model $p_x(x; \theta)$ and a target distribution $p^*_x(x)$ can be written:

$$\begin{aligned} \mathcal{L}^{forward}_{bg}(\theta) &= D_{KL}\left[p^*_x(x) \,\|\, p_x(x;\theta)\right] \\ &= -\mathbb{E}_{p^*_x(x)}\left[\log p_x(x;\theta)\right] + const. \end{aligned} \tag{3}$$

Having access to a set of training samples $\{x_n\}_{n=1}^N$ from $p^*_x(x)$, the expectation over $p^*_x(x)$ can be estimated by Monte Carlo Papamakarios et al. (2021). For estimating the background scene, the first term in the RHS of Eq.2 becomes zero ($p_{z_{fg}}(z_{fg}) = 0$). By inserting that into Eq. 3 we obtain

$$\mathcal{L}^{forward}_{bg}(\theta) \approx -\frac{1}{N}\sum_{n=1}^{N} \log p_{z_{bg}}(f^{-1}_{bg}(x_n)) - \log \left| detJ_{f^{-1}_{bg}}(x_n) \right| + const. \tag{4}$$

For evaluating Eq. 4, a set of samples $\{x_n\}_{n=1}^N$ that only contain the background scene are needed. This is usually not possible as foreground objects may exist in all training samples. For constraining the network to map nothing but the background features into z_{bg}, we additionally optimize the parameters during the backward flow by sampling $z_{bg} \sim N(0, 1)$ and reconstructing the background. By assembling the data tensors using random batch selection, the likelihood of having at least one background pixel in the tensor across the batch dimension increases. Thus, the network can generate the background scene through its reverse operations from $(z_{bg}, z_{fg}) = f^{-1}(x)$ computed in the forward pass by ignoring the foreground components. Using this idea, we formulate the backward reconstruction loss for the background scene over all samples in the batch B as follows.

$$\begin{aligned} &\hat{x}_{bg} = f(\hat{z}_{bg}, z_{fg} = \mathbf{O}), \quad \text{where} \quad \hat{z}_{bg} = z_{bg} + \epsilon \quad and \quad \epsilon \sim N(0, \sigma_\epsilon) \\ &\mathcal{L}^{backward}_{bg}(\theta) = \frac{1}{M}\sum_{j=1}^{M} \min_{\forall x \in B} \|x^j - \hat{x}^j_{bg}\|_2^2, \end{aligned} \tag{5}$$

where $\mathbf{O}$ is a null tensor being all zero and x^j denotes the j^{th} pixel of the sample x in the training batch. In Eq. 5, adding a small variation ϵ to the z_{bg} and then minimising the difference between the generated image with all the images in the batch for all M pixels, forces the z_{bg} to belong to the common part of the scene across the batch. In case that the training dataset includes diverse backgrounds, for instance images captured by multiple cameras with different poses, Eq. 5 is still valid if the training batch is sampled from the same background scene.

The background loss is computed by aggregating the losses of forward and backward flows:

$$\mathcal{L}_{bg}(\theta) = \mathcal{L}^{forward}_{bg} + \mathcal{L}^{backward}_{bg} \tag{6}$$

3.1.3 Foreground Generative Flow

After splitting the latent tensor z into two parts along the channel dimension and assigning one to the background z_{bg}, the other part is further transformed through four invertible blocks to generate the foreground component z_{fg} that represents the objects in the scene. Similar to SPAIR Crawford & Pineau (2019), we consider z_{fg} as structured variable. Given color images of size $H \times W \times 3$, the variable z_{fg} has a lower spatial resolution due to the squeeze layer of the network. For instance, with one squeeze layer, z_{fg} is a feature map of size $h \times w \times c$, where $h = \frac{1}{2}H$ and $w = \frac{1}{2}W$. Afterwards, the z_{fg} is partitioned into four parts along its channel dimension by applying a split operator. Each part is transformed through four invertible blocks (i.e. head branches) and yields a set of latent maps. Hence, each cell with indices (i, j), $i \in \{0, ..., h-1\}$, $j \in \{0, ..., w-1\}$ and of length c in the output of size $h \times w$ is associated with a set of $(z_{pres}, z_{scale}, z_{loc}, z_{what})$ variables.

Note that the variable $z_{pres} \in \{0, 1\}$ represents whether a specific cell belongs to the foreground or not. For foreground cells ($z_{pres} = 1$), the variables $z_{scale} \in \mathbb{R}^2$ and $z_{loc} \in \mathbb{R}^2$ represent the size and displacement vector to the center of the object. The variable $z_{what} \in \mathbb{R}^{c-5}$ corresponds to other object attributes like appearance and color for visual data. We impose base distributions to each of these structured variables.

The binary variable z_{pres} is modeled by a Bernoulli distribution using Gumbel-Softmax $z^{ij}_{pres} \sim RelaxedBernoulli(\beta^{ij}_{pres})$ Jang et al. (2017). See the Appendix for more details. We impose

Gaussian priors for the latent variables z_{scale} and z_{loc}. Since z_{loc}^{ij} represents the relative displacement vector of cell (i, j) from the closest object in the image, we impose a prior $z_{loc} \sim N(0, 1)$ to the normalized displacement vectors. Here, unit one means the diagonal size of the image. The scale of objects are configured based on normalized mean and variance of object sizes in images by imposing a prior $z_{scale} \sim N(\mu_{scale}, \sigma_{scale})$. We assign a bimodal Gaussian distribution to z_{what} as it represents the attributes of each cell depending on whether it belongs to the background or foreground. Depending on whether the cell (i, j) contains an object or not, it is sampled from one of two Gaussian distributions:

$$z_{what} \sim \begin{cases} N(\mu_{fg}, \sigma_{fg}), & \text{if } z_{pres} = 1 \\ N(\mu_{bg}, \sigma_{bg}), & \text{otherwise} \end{cases}$$

The base distribution on the foreground components z_{fg} can be defined as:

$$p_{z_{fg}}(z) = \prod_{ij} p(z_{pres}^{ij}) \cdot \Big(p(z_{scale}^{ij}) \cdot p(z_{loc}^{ij}) \cdot p(z_{what}^{ij}) \Big)^{p_{pres}^{ij}}. \tag{7}$$

Consequently, we can write:

$$\log p_{z_{fg}}(z) = \sum_{ij} \Big[\log p(z_{pres}^{ij}) + p(z_{pres}^{ij}) \cdot \Big(\log p(z_{scale}^{ij}) + \log p(z_{loc}^{ij}) + \log p(z_{what}^{ij}) \Big) \Big]. \tag{8}$$

Similar to what we showed earlier for the background components, we optimize the parameters of DeNF both via its forward and backward flows. As in Eq.4, we compute the forward loss for the foreground components by using Eq. 8 during the forward flow by having access to the training data:

$$\mathcal{L}_{fg}^{forward}(\theta) \approx -\frac{1}{N} \sum_{n=1}^{N} \log p_{z_{fg}}(f_{fg}^{-1}(x_n)) - \log \Big| detJ_{f_{fg}^{-1}}(x_n) \Big| + const. \tag{9}$$

In the backward flow, we sample from the model by considering only the foreground components for reconstructing the image. The reconstructed image is compared with input images for those regions where objects are present ($z_{pres} = 1$). So the backward loss can be written as:

$$\begin{aligned} &\hat{x}_{fg} = f_\theta(z_{pres}, z_{scale}, z_{loc}, z_{pres} \circ z_{what}, z_{bg} = \mathbf{O}), \\ &\hat{x} = z_{pres} \cdot \hat{x}_{fg} + (1 - z_{pres}) \cdot \hat{x}_{bg}, \\ &\mathcal{L}_{fg}^{backward}(\theta) = \|x - \hat{x}\|_2^2, \end{aligned} \tag{10}$$

Here, the symbol $\circ$ denotes the Hadamard product. By minimizing both forward and backward losses (Eq. 9 and 10), the model learns to decompose the representation of objects into a set of structured latent variables and to generate the images reversely.

$$\mathcal{L}_{fg}(\theta) = \mathcal{L}_{fg}^{forward} + \mathcal{L}_{fg}^{backward} \tag{11}$$

3.1.4 DISENTANGLEMENT OF STRUCTURED LATENT VARIABLES

For disentangling the representations of the objects via structured latent variables, two constraints are applied. The first constraint is imposing two different prior distributions on the elements of z_{what}, conditioned on their corresponding z_{pres}. To maximize the likelihood of foreground and background elements on the z_{what}, the model explores generating a binary map for z_{pres} that masks the objects in the images.

The second constraint is the prediction of z_{pres} from the (z_{loc}, z_{scale}) variables. According to our definitions of the structured latent variables, we expect that for cells (i, j) belonging to the foreground, the pair of vectors $(z_{scale}^{ij}, z_{loc}^{ij})$ indicates the bounding box of the object that cell (i, j) belongs to. Hence, rendering a fixed-size glimpse of unity values (i.e. matrix of ones) inside the detected bounding boxes and zero values outside should partially resemble the binary map of z_{pres} corresponding with the cell (i, j). In order to construct a differentiable binary object mask, a spatial transformer ($\mathcal{T}$) Jaderberg et al. (2015) is used. The spatial transformer (ST) can be considered as an adaptive spatial sampler from a map. It receives the bounding box coordinates, interpolates inside the bounding box in a given map, and returns the re-sampled region with the specified dimension.

Applying the ST in a reverse mode ($\mathcal{T}^{-1}$), scales and locates a given glimpse of unity values into a map initialized by zero values. Generating such a mask for all foreground cells based on (z_{loc}, z_{scale}) values and superposing them yields a binary image, called $\hat{z}_{pres}$. Contingent upon whether the representation of objects' scale and location are in agreement with their presence map, the $\hat{z}_{pres}$ should resemble z_{pres}. This results in introducing a compatibility loss as follows.

$$\mathcal{L}_{com}(\theta) = \frac{\sum_{ij}(\hat{z}^{ij}_{pres} - z^{ij}_{pres})^2}{\sum_{ij}\hat{z}^{ij}_{pres} + \sum_{ij} z^{ij}_{pres}}, \quad \text{where} \quad \hat{z}^{ij}_{pres} = min\big(1, \sum_{ij} \mathcal{T}^{-1}(\mathbb{1}, z^{ij}_{scale}, z^{ij}_{loc})\big). \quad (12)$$

Here, $\mathbb{1}$ denotes a matrix of ones and the minimum operator clamps all elements of superposed binary maps into the range [0, 1].

3.1.5 Inference and Training

The learning parameters of DeNF are optimized through the background and foreground generative flows by simultaneously minimizing Eqs. 6, 11, and 12 using gradient descent.

$$\mathcal{L}_{total}(\theta) = \mathcal{L}_{bg} + \mathcal{L}_{fg} + \mathcal{L}_{comp}. \quad (13)$$

We provide pseudo code of the training procedure, details of the proposed architecture and the hyperparameters in the supplementary material.

Scene reconstruction with single object: After training the model, we can generate the scene by preserving only one specific object and masking the others. During inference, the pair of vectors (z_{scale}, z_{loc}) of each cell that contains any object ($z_{pres} = 1$) represents a bounding box in the scene. For detecting how many objects are in the scene, we apply a non-maximum suppression algorithm on all detected bounding boxes to have a single bounding box per object. For masking other objects, we construct a binary mask that is identical with z_{pres} inside the target object bounding box and zero elsewhere. By mixing x_{bg} and x_{fg} similar to what has been shown in Eq.10, the scene can be generated while preserving only a single object.

4 Experiments

4.1 Experimental setup and evaluation metrics

We evaluate our proposed model on two visual scene datasets. We compare DeNF with recent, competitive methods based both on VAEs and GANs. For evaluating the performance on a scene decomposition task, we use CLEVR Johnson et al. (2017), a data set of 3D rendered objects, as it is widely used for object-centric scene representation. Similar to prior works Monnier et al. (2021); Greff et al. (2019), we measure the similarity between ground-truth (instance) segmentations and our predicted object masks using the Adjusted Rand Index (ARI) Rand (1971); Hubert & Arabie (1985) to quantify segmentation quality. ARI is a measure of cluster similarity that can handle arbitrary permutations of the clusters.

For evaluating the performance of DeNF on real scene images, we compare the proposed model with both VAE and GAN-based models on the scene generation task with respect to the Fréchet Inception Distance (FID) Heusel et al. (2017). In this experiment, DeNF decomposes the input scene into the object instances and space (i.e. background). Then, by sampling from the foreground latent variables, it generates new examples of the input scene by preserving only a subset of objects.

4.2 Datasets

CLEVR data Johnson et al. (2017): This dataset contains $70,000$ images. They have a resolution of 240×320 pixels but are scaled to 128×128 pixels following˜Greff et al. (2019). Each scene contains between three and ten objects, characterized in terms of shape (cube, cylinder, or sphere), size (small or large), material (rubber or metal), color (8 different colors), position (continuous), and rotation (continuous). The subset of images which contain 1-6 objects (inclusive) is used in our experiments; we refer to it as CLEVR6.

Table 1: Average foreground ARI (higher is better) on CLEVR6, compared with values reported in the literature.

Method	Framework	Metric
IODINE Greff et al. (2019)	VAE+RNN	98.8 ± 0.0
MONet Burgess et al. (2019)	VAE+RNN+CNN	96.2 ± 0.6
Slot Att. Locatello et al. (2020)	Transformer	98.8 ± 0.3
ObSuRF Stelzner et al. (2021)	NeRF	98.3 ± 0.8
Sprites Monnier et al. (2021)	DTI	97.2 ± 0.2
DeNF (ours)	NF	96.3 ± 0.2

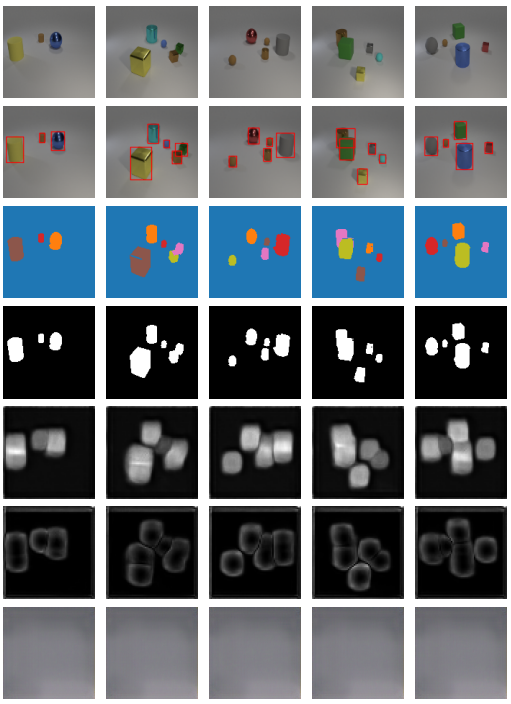

Figure 3: Visualized examples of CLEVR6 data. The rows from top to bottom show the original images, detected objects, segmented foreground, z_{pres} map, norm of z_{scale}, norm of z_{loc}, and the generated background.

Table 2: FID score (lower is better) on the Real-Traffic data sets, compared with values reported in the literature.

Method	Metric
GENESIS Engelcke et al. (2019)	167.1
BlockGAN Phuoc et al. (2020)	57.9
DCGAN Radford et al. (2016)	47.6
RELATE Ehrhardt et al. (2020)	42.0
DRAGAN Kodali et al. (2017)	38.8
DeNF (ours)	**25.6**

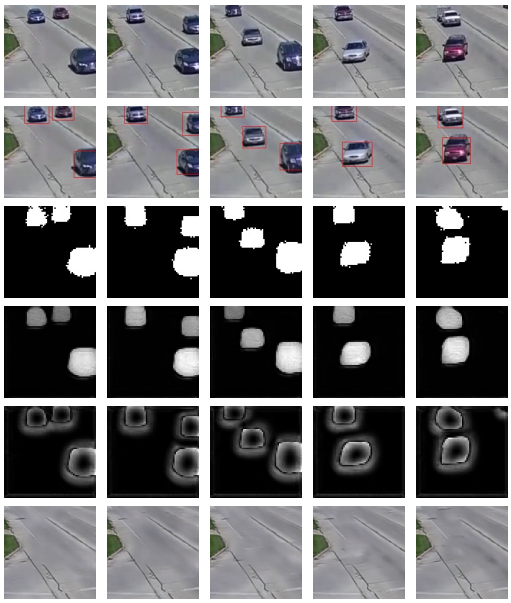

Figure 4: Visualized examples of Real-Traffic data; The rows from top to bottom show the original images, detected bounding boxes, z_{pres} map, norm of z_{scale}, norm of z_{loc}, and the generated background.

Real-Traffic data Ehrhardt et al. (2020) is a publicly available real-world image data, including five hours of footage of a busy street intersection, divided into fragments containing from one to six cars. All images are resized to have a size of 128×128 pixels. Similar to Ehrhardt et al. (2020), we keep the images of 560 videos as training set and the images of 123 videos for testing.

4.3 Results

Object Instance Segmentation. For the object instance segmentation task, we compare DeNF with state-of-the-arts on the CLEVR6 dataset. Table 1 shows the ARI score for different methods. DeNF shows competitive performance in term of the ARI score. While some prior works such as IODINE and ObSURF show a slightly a higher score, their underlying RNN and NeRF architectures are expensive both in terms of computation and memory, limiting their practicality and utility. Differently, the proposed DeNF contains only 12 invertible blocks and a parallel spatial transformer that is scalable in terms of number of objects in the scene. Furthermore, while some of those methods present results on scene synthesis, they use simple scenes with sprites and a constant non-textured

background. In contrast, the proposed algorithm works also well in a hard generative task as will be shown in the following section.

Scene Image Generation. Figure 4 shows some input images of the Real-Traffic dataset along with the structured variables that DeNF learns. For comparing the quality of scene decomposition and generation, we report the FID score in Table 2. DeNF outperforms the best state-of-the-art method by a wide margin. The obtained higher score of DeNF can be explained by its fidelity to image contents due to its invertible mapping. To demonstrate this property, we show the predicted background scene for different images in Fig. 5. As can be seen, there are only minor differences across background context, for instance the pole shadows on the surface of the road. DeNF preserves the visual context in its backward background generative process that is challenging for VAE or GAN-based approaches. In supplementary material additional examples of scene reconstruction are shown.

Ablation experiments. We conduct some experiments for an ablation study. The first ablation reduces the spatial dimensions of latent structured variables from 64×64 into 32×32 cells by adding another squeeze operator before the last split block to the network. In the second ablation the binomial Gaussian prior of the z_{what} is replaced by a unimodal Gaussian distribution. It is worth mentioning that by removing the spatial transformer from the DeNF, the model does not work as it cannot disentangle the latent variables anymore. As shown in Table 3, by reducing the size of latent space, the ARI score drops. This is mainly due to representing small objects in a scene with only a few cells in the latent space that causes inaccurate bounding-box estimation and $\hat{z}_{pres}$ prediction. Imposing a Gaussian prior only to the foreground elements of z_{what} does not degrade the ARI score significantly. This small reduction in performance may happen since the background elements of z_{what} are masked (see Eq. 10) and filled by zeros in the backward process by sampling from the foreground. Therefore, the model learns to produce zero values for the background elements of z_{what} in the forward flow.

Figure 5: Examples of some generated backgrounds for the Real-Traffic dataset. DeNF learns to generate various background images (for instance with corresponding shadows on the road surface) with a high fidelity to the input images.

Table 3: Ablation study. ARI score (higher is better) on CLEVR6 data.

Experiment	ARI
smaller latent space (32×32 cells)	94.4
w/o bimodal prior	95.8
DeNF (full)	96.3

5 Discussion

Decomposable representations are an active area of research, being a key step towards compositionality. This work is a step towards that goal, showing the benefit of using normalizing flows for visual scene decomposition. The proposed unsupervised algorithm (DeNF) leverages recent advancements in normalizing flows to formulate the scene decomposition as a probabilistic modeling. DeNF consists of a mixture of bidirectional flows which decompose the scene into a set of structured latent variables. Due to its invertible neural architecture, it is able to reversely combine the extracted variables to generate the input scene with a high fidelity to the visual contents of the input scene. By experiments on synthetic and real image data, DeNF shows competitive performance, compared to other generative frameworks, based on VAE or GAN models. While it performs comparably with state-of-the-art methods in scene decomposition, those methods only use simplistic data for scene generation. In contrast, the proposed algorithm is evaluated on a challenging real-image dataset for scene generation and in terms of fidelity to the scene's visual contents, is superior to the best competing methods by a wide margin.

REFERENCES

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. MONet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Crawford, E. and Pineau, J. Spatially invariant unsupervised object detection with convolutional neural networks. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2019.

Dinh, L., Krueger, D., and Bengio, Y. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

Ehrhardt, S., Groth, O., Monszpart, A., Engelcke, M., Posner, I., Mitra, N., and Vedaldi, A. RELATE: Physically plausible multi-object scene synthesis using structured latent spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. GENESIS: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations (ICLR)*, 2019.

Engelcke, M., Jones, O. P., and Posner, I. GENESIS-V2: Inferring unordered object representations without iterative refinement. *arXiv preprint arXiv:2104.09958*, 2021.

Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning (ICML)*, 2019.

Greff, K., van Steenkiste, S., and Schmidhuber, J. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hubert, L. and Arabie, P. Comparing Partitions. *Journal of Classification*, 2(1):193–218, 1985.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On convergence and stability of GANs. *arXiv preprint arXiv:1705.07215*, 2017.

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations (ICLR)*, 2020.

Liu, Y., Qin, Z., Anwar, S., Caldwell, S., and Gedeon, T. Are deep neural architectures losing information? invertibility is indispensable. In *International Conference on Neural Information Processing (ICONIP)*, 2020.

Liu, Y., Qin, Z., Anwar, S., Ji, P., Kim, D., Caldwell, S., and Gedeon, T. Invertible denoising network: A light solution for real noise removal. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Maddison, C., Mnih, A., and Teh, Y. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*. International Conference on Learning Representations (ICLR), 2017.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020.

Monnier, T., Vincent, E., Ponce, J., and Aubry, M. Unsupervised layered image decomposition into object prototypes. *arXiv preprint arXiv:2104.14575*, 2021.

Niemeyer, M. and Geiger, A. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research (JMLR)*, 22(57):1–64, 2021.

Phuoc, T. N., Richardt, C., Mai, L., Yang, Y., and Mitra, N. J. BlockGAN: Learning 3d object-aware scene representations from unlabelled images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015.

Stelzner, K., Kersting, K., and Kosiorek, A. R. Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*, 2021.

van Steenkiste, S., Kurach, K., Schmidhuber, J., and Gelly, S. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309–325, 2020.

Zhu, W., Shen, Y., Yu, L., and Sanchez, L. P. A. GMAIR: Unsupervised object detection based on spatial attention and gaussian mixture. *arXiv preprint arXiv:2106.01722*, 2021.

– Supplementary Materials –

A IMPLEMENTATION DETAILS

In this section, we provide further details to ensure the reproducibility.

A.1 BERNOULLI PRIOR ON z_{pres}

The binary variable z_{pres} for a cell with indices (i, j) is modeled by a Bernoulli distribution using Gumbel-Softmax $z^{ij}_{pres} \sim RelaxedBernoulli(\beta^{ij}_{pres})$ Jang et al. (2017). A concrete distribution Jang et al. (2017); Maddison et al. (2017) with a relaxation to a categorical distribution based on Gumble-Softmax trick makes the base distribution differentiable during training. According to Maddison et al. (2017), the sampling procedure for z^{ij}_{pres} can be described as

$$z^{ij}_{pres} = \frac{1}{1 + \exp -\frac{\log \alpha + L}{\lambda}}, \tag{14}$$

where $L = \log U - \log(1 - U)$ and $U \sim Uniform(0, 1)$. The temperature parameter $\lambda \in (0, \infty)$ controls the degree of approximation. As $\lambda \to 0$, the z^{ij}_{pres} converges to Bernoulli with parameter $\frac{\alpha}{1+\alpha}$.

A.2 PSEUDO CODE OF TRAINING DENF

Algorithm 1 describes the steps of training the DeNF.

Algorithm 1 Training procedure of DeNF

Input $x \in \{x\}_{n=1}^{N}$
Output Parameters of network (θ)

while $t \leq Max$ **do**
  $(z_{bg}, z_{fg}) = f_1^{-1}(x)$
  $(z^{logit}_{pres}, z_{scale}, z_{loc}, z_{what}) = f_2^{-1}(z_{fg})$
  $\hat{x}_{bg} = f_1(z_{bg} + N(0, \epsilon), z_{fg} = \mathbf{O})$
  $z_{pres} \sim$ Bernoulli$(z^{logit}_{pres}, \lambda)$
  $\hat{z}_{fg} = f_2(z_{pres}, z_{scale}, z_{loc}, z_{pres} \circ z_{what})$
  $\hat{x}_{fg} = f_1(\hat{z}_{fg}, z_{bg} = \mathbf{O})$
  $\hat{x} = z_{pres} \cdot \hat{x}_{fg} + (1 - z_{pres}) \cdot \hat{x}_{bg}$ ▷ Reconstruct input image
  $\hat{z}_{pres} = min\big(1, \sum_{ij} \mathcal{T}^{-1}(\mathbb{1}, z^{ij}_{scale}, z^{ij}_{loc})\big)$ ▷ Predicting z_{pres} using ST

  $\mathcal{L}^{forward}_{bg} \leftarrow -$log-likelihood$(p_{z_{bg}})$ ▷ Eq. 4
  $\mathcal{L}^{backward}_{bg} \leftarrow (x, \hat{x}_{bg})$ ▷ Eq. 5
  $\mathcal{L}^{forward}_{fg} \leftarrow -$log-likelihood$(p_{z_{fg}})$ ▷ Eq. 8
  $\mathcal{L}^{backward}_{fg} \leftarrow (x, \hat{x})$ ▷ Eq. 10
  $\mathcal{L}_{com} \leftarrow (\hat{z}_{pres}, z_{pres})$ ▷ Eq. 12
  $\mathcal{L}_{total} \leftarrow (\mathcal{L}_{bg}, \mathcal{L}_{fg}, \mathcal{L}_{com})$ ▷ Eq. 13
  $\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta \frac{\partial \mathcal{L}_{total}}{\partial \theta}$ ▷ update
end while

A.3 ARCHITECTURE DETAILS

For implementing the invertible blocks in DeNF, we use coupling layers Dinh et al. (2016). We followed the same choice of neural architecture for coupling layers as Liu et al. (2021) with residual convolutional operators.

Table 4: Architecture details of DeNF

Layer name	Layer type	Input size	output size	Kernel size	Activation
Coupling 1	Residual	$128 \times 128 \times 3$	$128 \times 128 \times 3$	3×3	LeakyReLU
Coupling 2	Residual	$128 \times 128 \times 3$	$128 \times 128 \times 3$	3×3	LeakyReLU
Coupling 3	Residual	$128 \times 128 \times 3$	$128 \times 128 \times 3$	3×3	LeakyReLU
Coupling 4	Residual	$128 \times 128 \times 3$	$128 \times 128 \times 3$	3×3	LeakyReLU
Squeeze	-	$128 \times 128 \times 3$	$64 \times 64 \times 12$	-	-
Split	-	$64 \times 64 \times 12$	$64 \times 64 \times [1, 11]$	-	-
Coupling 5	Residual	$64 \times 64 \times 11$	$64 \times 64 \times 11$	3×3	LeakyReLU
Coupling 6	Residual	$64 \times 64 \times 11$	$64 \times 64 \times 11$	3×3	LeakyReLU
Coupling 7	Residual	$64 \times 64 \times 11$	$64 \times 64 \times 11$	3×3	LeakyReLU
Coupling 8	Residual	$64 \times 64 \times 11$	$64 \times 64 \times 11$	3×3	LeakyReLU
Split	-	$64 \times 64 \times 11$	$64 \times 64 \times [2, 2, 2, 5]$	-	-
Coupling 9	Residual	$64 \times 64 \times k$	$64 \times 64 \times k$	3×3	LeakyReLU
Coupling 10	Residual	$64 \times 64 \times k$	$64 \times 64 \times k$	3×3	LeakyReLU
Coupling 11	Residual	$64 \times 64 \times k$	$64 \times 64 \times k$	3×3	LeakyReLU
Coupling 12	Residual	$64 \times 64 \times k$	$64 \times 64 \times k$	3×3	LeakyReLU

A.4 Training hyperparameters

For both experiments on the Clevr and Real-Traffic data sets, images are resized to a size of 128×128 pixels. We use a batch size of 12. We use a learning rate (η) of $1\mathrm{e}{-4}$ with an Adam optimizer and gradient clipping with a maximum norm of 1.0. For quantitative results, DeNF is trained up to 20 epochs. Other hyperparameters are shown in Table 5. The temperature z_{pres} was linearly reduced starting at epoch 3.

Table 5: DeNF hyperparameters

Parameter	Symbol	Value (Clevr exp.)	Value (Real-Traffic exp.)
z_{bg} prior	(μ_{bg}, σ_{bg})	(0, 1.0)	(0, 1.0)
Additive noise to $p(z_{bg})$	σ_ϵ	0.2	0.2
z_{pres} temperature	λ	$0.1 \rightarrow 0.01$	$0.1 \rightarrow 0.01$
z_{loc} prior	$(\mu_{loc}, \sigma_{loc})$	(0, 1.0)	(0, 1.0)
z_{scale} prior	$(\mu_{scale}, \sigma_{scale})$	(0.20, 0.1)	(0.25, 0.1)
z_{what} prior for foreground	(μ_{fg}, σ_{fg})	(1.0, 0.1)	(1.0, 0.1)
z_{what} prior for background	(μ_{bg}, σ_{bg})	(0.0, 0.1)	(0.0, 0.1)

A.5 Evaluation details

For computing the ARI scores on the CLEVR data, similar to previous methods, we measured the instance segmentation quality by treating each foreground pixel and ignored the background region and its segmentation as cluster assignment. For computing the FID scores, 10,000 samples from DeNF were drawn which have been compared against the same number of images drawn from the test set.

A.6 Infrastructure and framework

All experiments were done by using PyTorch 1.10 on a single NVIDIA Tesla V100 GPU.

B DECOMPOSITION RESULTS

As described in section 3.1.5, the model is able to decompose and reconstruct the scene by preserving only one object at a time. Figure 6 visualizes some examples of scene reconstruction.

Figure 6: Visualized scene reconstruction with a single object using DeNF. The first column shows the original images. Other columns show the decomposed images.