Supplementary Material for OneGAN: Simultaneous Unsupervised Learning of Conditional Image Generation, Foreground Segmentation, and Fine-Grained Clustering

A Summary of notation

For convenience, in Tab. 1 we provide a complete listing of the notation used in our paper.

B Regularization

Due to lack of space in the main text, we include the regularization loss terms as part of the supplementary.

During generation, we apply regularization on the latent vectors and on the mask image. The former serves to bound the range of the values to be close to the axis center and to be closely grouped.

$$\mathcal{L}_{R_v} = ||v_p||_2^2 + ||v_c||_2^2 + ||v_{bg}||_2^2$$

The regularization on the mask serves to direct the model to utilize the mask efficiently, with a balanced and decisive representation of background and foreground. For mask $I_m \in [0,1]^{H,W}$, with H, W the height and width of the mask. The first regularization term balances the mask value around the value of half.

$$\mathcal{L}_{M_B} = |\frac{1}{HW}(\sum_{i,j}^{HW}(I_m)_{i,j}) - \frac{1}{2}|$$

The second regularization term aims to make the masks more decisive. It is better described when the mask is between [-1,1], so we define $I'_m = 2 \cdot I_m - 1$. In the ideal case, all pixels are either 1 or -1 (either background of foreground), therefore, if we assume a balanced distribution, for each mask the average value of $\max(0, I'_m)$ is 0.5 and of $\min(0, I'_m)$ it's -0.5, since half of the pixels are zeroed in each term. This is the decisiveness regularization.

$$\mathcal{L}_{M_D} = |\frac{1}{HW}(\sum_{i,j}^{HW}\max(0,(I'_m)_{i,j})) - \frac{1}{2}| + |\frac{1}{HW}(\sum_{i,j}^{HW}\min(0,(I'_m)_{i,j})) + \frac{1}{2}|)$$

Together, the mask regularization loss is:

$$\mathcal{L}_{R_M} = \mathcal{L}_{M_B} + 0.1 \cdot \mathcal{L}_{M_D}$$

C Sub-networks architecture

In this section, we describe the details of each sub-network described in the main paper. The layers of each sub-network are listed in the tables Tab. 3–9, with some modules that are frequently used listed in Tab. 2. The majority of the networks are sequential. When more complicated connections are present, the input and output notations are there to guide the flow.

D Additional illustrations

D.1 Conditional generation

We supply more conditionally generated images to further demonstrate the conditional generation performance. We use generation conditioned on both very different classes to highlight the broad coverage of the representation and very similar classes to show the high sensitivity to detail.

In Fig. 1, the images are obtained by generating five different images per reference image in the top row. To achieve these results, our model has to perform two tasks. First, it has to be able to detect the child and parent classes under which the object is represented. Second, it needs to be able to generate a similar looking object with the predicted classes. The success in this task is evidence for both the generation and clustering capabilities of the model.

In the figure, each column shows a real image, followed by five generated images conditioned on the first image in respect to category. Additionally, in each row, all images are generated with the same z, showing how non-categorical information is consistent across the different categories and how the pose is disentangled from the shape category.

For both birds and dogs, we can see that the generated images have very similar properties to their conditioned image. In both cases, we can see the large coverage of different classes and the fine detailed differences between similar classes. For cars, we can see that that the generated images apply the same color, but the car shape is changing, indicating that the model has categorized the cars by their color and not by their model.

D.2 Reconstruction

We supply more images to show the reconstruction path with the resulting reconstructed images, reconstructed backgrounds, and segmentation masks.

In Fig. 2, the results show the images generated by the reconstruction process. The generated mask shows the model's ability to detect and segment the object, the background image shows the model's ability to repaint the background, and the foreground image shows the model's ability to detect and reconstruct conditioned on the object class.

The segmentation works under many different poses, sizes, and backgrounds. For dogs and cars, it can be seen that our model is sometimes better than

the "ground-truth" masks, which were generated by a pre-trained network. The background repainting works well in the majority of cases, but we do notice that some backgrounds work better than others. The challenges are mostly noticeable in the cars dataset, where there was the smallest amount of background patches available in X_{bg}, leading to a less powerful background discriminator. Subsequently, the performance of the background generation was affected.

D.3 Image to image translation

We supply more images, Fig. 3, to show the image to image translation capability of the model. We show that the disentanglement that emerged from the design allows manipulation of the reconstructed image by replacing the child code with a code from an arbitrary category. We can see that not only do the objects in the images change appearance, but the change is consistent across different images, while the background is mostly unaffected.

In birds, we can see that the generator is usually able to detect the different parts of the birds (wings, head, beak) and apply the correct color manipulation to the correct area. Furthermore, the color manipulation works on birds of different shapes and different original colors. The background is sometimes slightly altered, first, because it is regenerated every time, and second, because the foreground mask is soft and sometimes applies a slight manipulation on the background as well. In dogs, the manipulation is less effective, but it is noticeable and is correlated to the applied category. In cars, we can see the color manipulation for many different colors. We can also notice how the background is mostly unchanged and that the manipulation is applied correctly on the car chassis and not on other parts like windows, tires and lights.

D.4 Background inpainting

Due to space limits, we could not address all tasks in the main paper. As an intermediate step of the reconstruction task, our model also performs a side-task of object foreground extraction and background inpainting. The model first detects the foreground in the image and produces a segmentation mask. Then, with the mask, the background is encoded and reconstructed. Because the mask is a prediction and not a ground-truth, the model cannot only fill the masked pixels with background texture, but has to assume that the mask was not perfect and reconstruct the entire image. The drawback of this method is that the background is not always identical to the source in the background area, but the benefit is that the object is fully removed even when it is not fully covered by the mask.

We compare our model against images produced with Deep-Image-Prior (DIP; Ulyanov et al., CVPR 2018). There are two variants. In the fist, DIP receives the ground-truth mask and in the second, the predicted mask is given. DIP optimizes its network for 1000 steps on the input image.

The results can be seen in Fig. 4. It can be observed that DIP works relatively well when using a perfectly covering mask, but fails when the mask is not perfect

Table 1. The components of the OneGAN model

	Symbol	Description	Computed as (or a comments)
Variables	$\phi_c \in [1, N_C]$	child class	
	$\phi_p \in [1, N_P]$	parent class	
	$e_c \in \{0,1\}^{N_C}$	child class one-hot vector (style)	$e_c[i] = \delta_{i,\phi_c}$
	$e_p \in \{0,1\}^{N_P}$	parent class one-hot vector (shape)	$e_p[i] = \delta_{i,\phi_p}$
	$e_{bg} \in \{0,1\}^{N_P}$	background one-hot vector	$e_{bg} = e_p$
	$z \in \mathbb{R}^{d_z}$	pose code	$z[i] \sim \mathcal{N}(0,1)$
	$v_c \in \mathbb{R}^{d_c}$	style code vector	$v_c = V_{c_0}(e_c)$
	$v_p \in \mathbb{R}^{d_p}$	shape code vector	$v_p = V_{p_0}(e_p)$
	$v_{bg} \in \mathbb{R}^{d_{bg}}$	background code vector	$v_{bg} = V_{bg_0}(e_{bg})$
	A_{fg}	foreground pre-image	$A_{fg} = G_{fg_0}(v_p, z)$
	A_{bg}	background pre-image	$A_{bg} = G_{bg_0}(v_{bg}, z)$
	I_m	foreground mask	
	I_{fg}	foreground image	$(I_{fg}, I_m) = G_{fg_2}(G_{fg_1}(A_{fg}, v_p), v_c)$
	I_{bg}	background image	$I_{bg} = G_{bg_1}(A_{bg})$
	I	full image	$I = I_{bg} \circ (1 - I_m) + I_{fg} \circ I_m$
	B_{fg}	foreground bypass	$B_{fg} = E_{p_1}(I)$
	B_{bg}	background bypass	$B_{bg} = E_{bg_1}(I, I_m)$
	X_c	image domain	
	X_{bg}	background image domain	
Networks	V_c	embedding LUT of child class	
	V_p	embedding LUT of parent class	
	V_{bg}	embedding LUT of background	
	G_{fg}	foreground generator	$G_{fg_2}(G_{fg_1}(G_{fg_0}(v_p, z), v_p), v_c)$
	G_{bg}	background generator	$G_{bg_1}(G_{bg_0}(v_{bg}, z))$
	E_c	style encoder	$E_c(I)$
	E_p	content encoder	$E_p(I)$
	E_{bg}	content encoder	$E_{bg}(I, I_m)$
	D_c	image discriminator	
	D_{bg}	background discriminator	
Parameters	N_C	number of child classes	Depends on the dataset.
	N_P	number of parent classes	$N_P < N_C$, Depends on the dataset.
	d_z	dimensionality of pose code	$d_z = 100$
	d_c	dimensionality of style code	$d_c = 32$
	d_p	dimensionality of shape code	$d_p = 16$
	d_{bg}	dimensionality of background code	$d_{bg} = 32$
	H, W	size of image	$H = W = 128$

and does not fully cover the object. In contrast, our model suffers less when the mask is not perfect. We can also see that our model does not exactly inpaints the background but actually repaints it, which usually results in a slightly different background even where the image was not masked, but as we mentioned above, it may be beneficial when the mask is not perfect. Finally, our model performs the inpainting task in a single forward path instead of 1000 iterations of DIP.

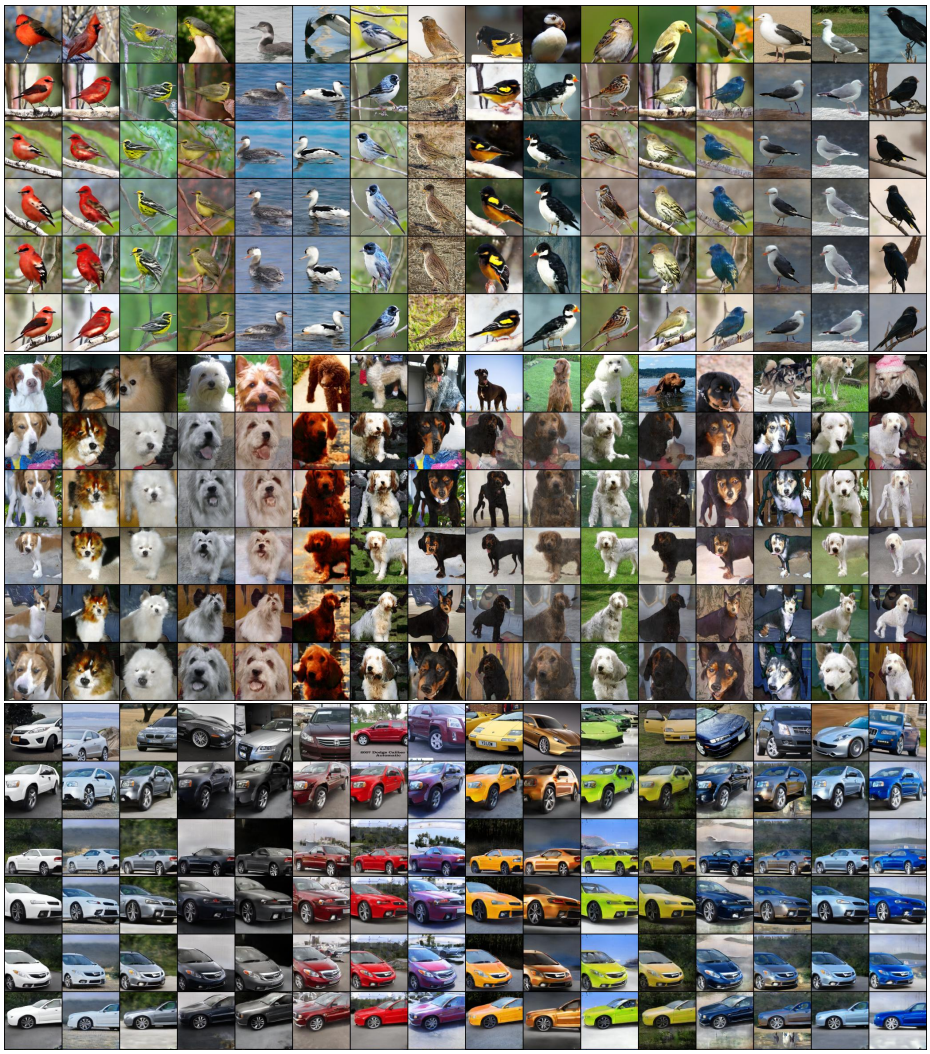

Fig. 1. Conditional Image Generation. From top to bottom: (i) real image, (ii-vi) generation of images with the encoded parent and child codes and a different vector z per row.

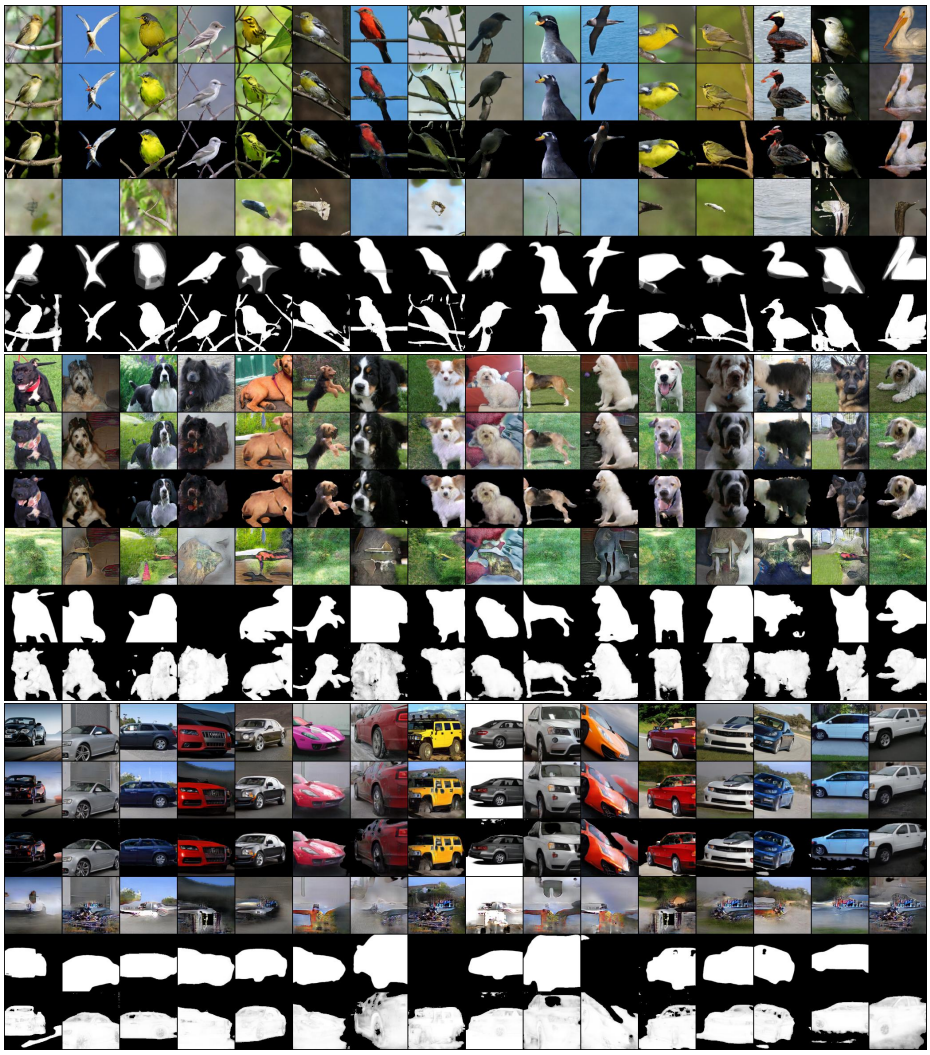

Fig. 2. Image Reconstruction. From top to bottom: (i) real image, (ii) reconstructed image, (iii) reconstructed foreground, (iv) reconstructed background, (v) ground-truth foreground mask, (vi) predicted foreground mask.

Fig. 3. Image to Image Translation. From left to right: (i) real image, (ii-xiii) reconstructed images when the child code e_c in each column is switched with a code from a selected category represented by the top image.

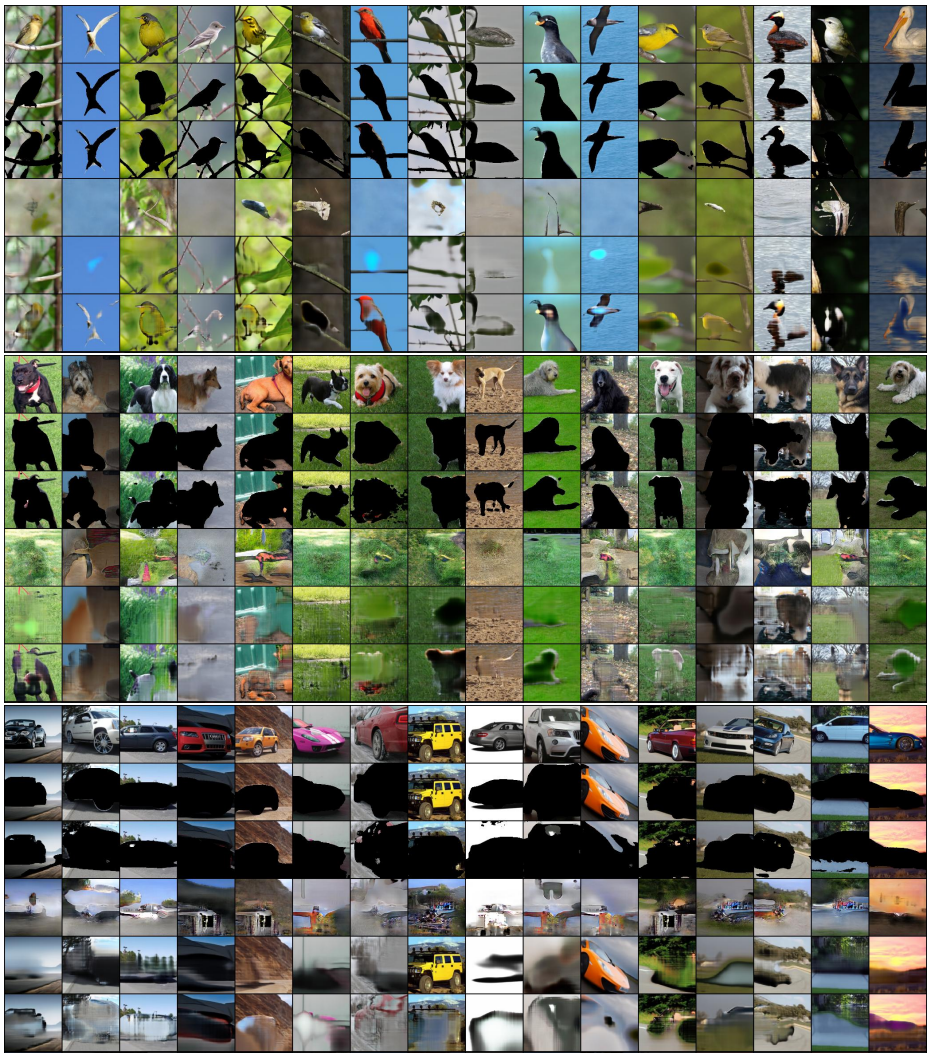

Fig. 4. Background Inpainting. From top to bottom: (i) original image, (ii) image masked with real mask, (iii) image masked with predicted mask, (iv) OneGAN, (v) DIP with real mask, (vi) DIP with predicted mask.

Table 2. General modules

Module	layers	input	output
GLU-LNorm	ChannelSplit	x	x_L, x_R
	LayerNorm	x_L	x'_L
	Sigmoid	x_R	x'_R
	Multiply	x'_L, x'_R	-
UPBlk (c_i, c_o, S)	Upsample2d($S/2$, S)	-	-
	K3P1Conv2d($c_i, 2c_o$)	-	-
	GLU-LNorm	-	-
DOWNBlk (c_i, c_o)	K4S2P1Conv2d(c_i,c_o)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
RESBlk0 (c_i)	K3P1Conv2d(c_i,$2c_i$)	x	-
	GLU-LNorm	-	-
	K3P1Conv2d(c_i,$2c_i$)	-	-
	GLU-LNorm	-	d
	Add	x, d	-
RESBlk (c_i, d, c_o)	K3P1Conv2d($c_i + d$,$2c_i$)	-	-
	GLU-LNorm	-	-
	RESBlk0(c_i)	-	-
	RESBlk0(c_i)	-	-
	K3P1Conv2d(c_i,$2c_o$)	-	-
	GLU-LNorm	-	-

Table 3. Background Generator G_{bg}

Module	layers	input	output
V_{bg}	Linear(N_P, d_{bg})	e_{bg}	v_{bg}
G_{bg_0}	Linear($d_{bg} + d_z$, 32768)	v_{bg}, z	-
	Reshape(2048,4,4)	-	-
	GLU-LNorm	-	-
	UPBlk(1024,512,8)	-	-
	UPBlk(512,256,16)	-	A_{bg}
G_{bg_1}	UPBlk(256,128,32)	A_{bg}	-
	UPBlk(128,64,64)	-	-
	UPBlk(64,32,128)	-	-
	K3P1Conv2d(32,3) + tanh	-	I_{bg}

Table 4. Foreground Generator G_{fg}

Module	layers	input	output
V_p	Linear(N_P, d_p)	e_p	v_p
V_c	Linear(N_C, d_c)	e_c	v_c
G_{fg_0}	Linear($d_p + d_z$, 32768)	v_p, z	-
	Reshape(2048,4,4)	-	-
	GLU-LNorm	-	-
	UPBlk(1024,512,8)	-	-
	UPBlk(512,256,16)	-	A_{fg}
G_{fg_1}	UPBlk(256,128,32)	A_{fg}	-
	UPBlk(128,64,64)	-	-
	UPBlk(64,3,128)	-	C_{fg_0}
G_{fg_2}	RESBlk(64,d_p,32)	C_{fg_0}, v_p	C_{fg_1}
	RESBlk(32,d_c,16)	C_{fg_1}, v_c	C_{fg_2}
	K3P1Conv2d(16,3) + tanh	C_{fg_2}	I_{fg}
	K3P1Conv2d(16,1) + sigmoid	C_{fg_2}	I_m

Table 5. Style Encoder E_c

Module	layers	input	output
E_{c_1}	K4S2P1Conv2d(3, 64)	I	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	DOWNBlk(64, 128)	-	-
	DOWNBlk(128, 256)	-	H_c
E_{c_0}	DOWNBlk(256, 512)	H_c	-
	DOWNBlk(512, 1024)	-	-
	K3P1Conv2d(1024, 1024)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Reshape(16384)	-	-
	Linear(16384, 512)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	h_c
	Linear((512, N_C)	h_c	$\hat{e}_c$
	Linear(512, d_c)	h_c	μ_c
	Linear(512, d_c)	h_c	σ_c

Table 6. Shape Encoder E_p

Module	layers	input	output
E_{p_1}	K4S2P1Conv2d(3, 64)	I	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	DOWNBlk(64, 128)	-	-
	DOWNBlk(128, 256)	-	H_p
	K3P1Conv2d((256, 512)	H_p	-
	GLU-LNorm	-	-
	UPBlk(256,256)	-	B_{fg}
E_{p_0}	DOWNBlk(256, 512)	H_p	-
	DOWNBlk(512, 1024)	-	-
	K3P1Conv2d(1024, 1024)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Reshape(16384)	-	h
	Linear(16384, 512)	h	-
	LayerNorm	-	-
	lReLU(0.2)	-	h_p
	Linear((512, N_C)	h_p	$\hat{e}_p$
	Linear(512, d_c)	h_p	μ_p
	Linear(512, d_c)	h_p	σ_p
	Linear(16384, 512)	h	-
	LayerNorm	-	-
	lReLU(0.2)	-	h_z
	Linear(512, d_z)	h_z	μ_z
	Linear(512, d_z)	h_z	σ_z

Table 7. Background Encoder E_{bg}

Module	layers	input	output
E_{bg_1}	K4S2P1Conv2d(4, 64)	I, I_m	-
	DOWNBlk(64, 128)	-	-
	DOWNBlk(128, 256)	-	H_{bg}
	K3P1Conv2d((256, 512)	H_{bg}	-
	GLU-LNorm	-	-
	UPBlk(256,256)	-	B_{bg}

Table 8. Background Discriminator D_{bg}

Module	layers	input	output
D_{bg}	DownSample2d(128, 126)	I	-
	K4S2P0Conv2d(3, 64)	-	-
	lReLU(0.2)	-	-
	K4S2P0Conv2d(64, 128)	-	-
	lReLU(0.2)	-	-
	K4S4P0Conv2d(128, 256)	-	-
	lReLU(0.2)	-	$H = D_{bg_C}(I)$
	K4S1P0Conv2d(256,1)	H	$D_{bg_A}(I)$
	K4S1P0Conv2d(256,1)	H	$D_{bg_B}(I)$

Table 9. Object Discriminator D_c

Module	layers	input	output
D_c	K4S2P1Conv2d(3, 64)	I	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	DOWNBlk(64, 128)	-	-
	DOWNBlk(128, 256)	-	-
	DOWNBlk(256, 512)	-	$D_{c_C}(I)$
	DOWNBlk(512, 1024)	-	-
	K3P1Conv2d(1024, 1024)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Reshape(16384)	-	H
	Linear(16384, 512)	H	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Linear(512, 1)	-	$D_{c_A}(I)$
	Linear(16384, 512)	H	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Linear(512, N_C)	-	$D_{c_B}(I)$