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

Yaniv Benny<sup>1</sup> and Lior  $Wolf^{1,2}$ 

<sup>1</sup> Tel-Aviv University, Israel <sup>2</sup> Facebook AI Research

Abstract. We present a method for simultaneously learning, in an unsupervised manner, (i) a conditional image generator, (ii) foreground extraction and segmentation, (iii) clustering into a two-level class hierarchy, and (iv) object removal and background completion, all done without any use of annotation. The method combines a Generative Adversarial Network and a Variational Auto-Encoder, with multiple encoders, generators and discriminators, and benefits from solving all tasks at once. The input to the training scheme is a varied collection of unlabeled images from the same domain, as well as a set of background images without a foreground object. In addition, the image generator can mix the background from one image, with a foreground that is conditioned either on that of a second image or on the index of a desired cluster. The method obtains state of the art results in comparison to the literature methods, when compared to the current state of the art in each of the tasks.

# Introduction

We hypothesize that solving multiple unsupervised tasks together, enables one to improve on the performance of the best methods that solve each individually. The underlying motivation is that in unsupervised learning, the structure of the data is a key source of knowledge and each task exposes a different aspect of it. We advocate for solving the various tasks in phases, where easier tasks are addressed first, and the other tasks are introduced gradually, while constantly updating the solutions of the previous sets of tasks. The method consists of multiple networks that are trained end-to-end and side-by-side to solve multiple tasks. The method starts from learning background image synthesis and image generation of objects from a particular domain. It then advances to more complex tasks, such as clustering, semantic segmentation and object removal. Finally, we show the model's ability to perform image-to-image translation. The entire learning process is unsupervised, meaning that no annotated information is used. In particular, the method does not employ class labels, segmentation masks, bounding boxes, etc. However, it does require a separate set of clean background images, which are easy to obtain in many cases.

**Contributions** Beyond the conceptual novelty of a method that treats singlehandedly multiple unsupervised tasks, which were previously solved by individual methods, the method displays a host of technical novelties, including: (i) a novel architecture that supports multiple paths addressing multiple tasks, (ii) employing bypass paths that allow a smooth transition between autoencoding and generation based on a random seed, (iii) backpropagation through three paths in each iteration, (iv) mixup module, which applies interpolation between latent representations of the generation and reconstruction paths, and more. Due to each of these novelties, backed by the ablation studies, we obtain state of the art results compared to the literature methods in each of the individual tasks.

## 2 Related work

Since our work touches on many tasks, we focus the literature review on general concepts and on the most relevant work. Generative models are typically based on Generative Adversarial Networks [11] or Variational Auto-Encoders [18]. In addition, these two can be combined [21]. Conditional image generation conditions the output on an initial variable, most commonly, the target class. CGAN [23] and InfoGAN [7] proposed different methods to apply the condition on the discriminator. Our work is more similar to InfoGAN, since we do not use labeled data and the label is not linked to any real image and no conditional discriminator can be applied. The condition is maintained by a classifier that tries to predict the conditioned label and, as a result, forces the generator to condition the result on that label. Semantic segmentation deals with the classification of the image pixels based on their class labels. For the supervised setting Unet [24], DeepLab [4], DeepLabV3 [5], HRNet [30] have shown great performance leaps using a regression loss. For the unsupervised case, more creative solutions are considered. In [6,16,32,31,8,27,14,3] a variety of methods have been used including inpainting, learning feature representation, clustering or video frames comparison. In **Clustering**, deep learning methods are the current state of the art. JULE [33] and DEPICT [10], cluster based on a learned feature representation. IIC [14] trains a classifier directly.

The most similar approach to ours is FineGAN [26], which our generators and discriminators are based upon. However, there are many significant differences and additions: (i) We added a set of encoders, which are trained to support new tasks. (ii) While FineGAN employs one-hot input, our generators use coded input, which is important for our autoencoding path. (iii) We added a skip connection, followed by a mixup module that combines the bypass tensor with the pre-image tensor. The mixup also allows passing only one of the tensors, making either the bypass or the pre-image optional in each flow. (iv) We employ single foreground generator instead of FineGAN's double hierarchical design, where we have found one generator to be dominant and the second one redundant. (v) Our model uses layer normalization [1] instead of batch normalization, which better performs for large number of classes, small batch size, and alternating paths. (vi) We define a new normalization method for the generators, where GLU [9] activation layers were used as non-linear activations. (vii) We add many losses, regularization terms, and training techniques that were not used in FineGAN, many of which are completely novel, as far as we can ascertain. As a result, our work outperforms FineGAN in all tasks and is capable of performing new tasks that its predecessor could not handle.

Mixup [35] is a technique for applying a weighted sum between two latent variables in order to synthesize a new latent variable. We use it to merge different paths in the model by mixing latent variables that are part coded by the encoder and part produced by the lower levels of the generation. As far as we can ascertain, this is the first usage of mixup to merge information from different paths. Our mixup is applied to image reconstruction in four different locations.

### 3 Method

To solve the tasks of clustering, foreground segmentation, and conditional generation with minimal supervision, our method trains multiple neural networks side-by-side. Each task is solved by applying the networks in a task-dependent order. Similarly, the method is trained by applying the networks in two different paths in each iteration, each path is optimized with a specific set of losses.

The architecture of the compound network consists of two Architecture generators, three encoders and two discriminators. Fig. 1 and Fig. 2 illustrate the two training paths. In the generation path, the generators produce a synthetic image conditioned on selected code, the encoders then retrieve the latent code from the generated images. In the reconstruction path, the encoders code an input image into latent code which is used by the generators to reconstruct the image. The reconstruction path is applied twice in each iteration, once on real images and once on fake images from the generation path. The reconstruction of fake images adds multiple capabilities of self-supervision such as reconstructing the background and mask, which is not applicable with real images without additional supervision. The sub-networks are fully detailed in the supplementary. The generation is performed by merging the results of two sepa-Generators rate generators that run in parallel to produce the output image. One generator is dedicated for generating the background and the other for the foreground. The generators are conditioned on a two-level hierarchical categorization. Each category has a unique child class  $\phi_c$  and a parent class  $\phi_p$  shared by multiple child classes. These classes are represented by the one-hot vectors  $(e_c, e_p)$ . An additional background one-hot vector  $e_{bq}$  affects the generation of the background images. Since there is a tight coupling between the class of the object (water bird, tropical bird, etc.) and the expected background, the typology of the background follows the coarse hierarchy, i.e. the parent class. The generator architecture is influenced by.

$$e_c[i] = \delta_{i,\phi_c}, \quad e_p[i] = \delta_{i,\phi_p}, \quad e_{bg} = e_p \tag{1}$$

The generation starts by converting the one-hot vectors into code vectors using learned embeddings. Such an embedding is often used when working with



**Fig. 1.** Flow of the generation path. The generators decode the four priors  $(e_{bg}, e_p, e_c, z)$  and produce three separate images (foreground, background, mask) that are combined into the final image. The generated image is then coded by three encoders to retrieve the latent variables and priors.



Fig. 2. Flow of the reconstruction path. The same sub-networks are rearranged to perform image reconstruction. The image is coded with the shape and style encoders and then decoded by the foreground generator to produce the foreground image and mask. Then the background encoder and generator code the masked image and produces a background image. The output image combines the foreground and background images. The mixup modules, placed in four different locations, merge the encoders' predicted codes with intermediate stages of the generation, acting as a robust skip connection.

categorical values. A fourth vector z is sampled from a multi-variate gaussian distribution to represent non-categorical features.

$$v_{bg} = V_{bg}(e_{bg}), \quad v_p = V_p(e_p), \quad v_c = V_c(e_c), \quad z \sim \mathcal{N}(0, 1)^{d_z}$$
 (2)

The background generator  $G_{bg}$  receives the background vector  $v_{bg}$  and noise z and produces a background image  $I_{bg}$ . The foreground generator  $G_{fg}$  receives the parent vector  $e_p$ , child vector  $e_c$  and the same z used in the background generation and produces a foreground image  $I_{fg}$  and a foreground mask  $I_m$ . The generator is optimized such that all foreground images with the same  $e_p$  will

have the same object shape and all images with the same  $e_c$  will have a similar object appearance. The latent vector z is implicitly conditioned to represent all non-categorical information, such as pose, orientation, size, etc. It is used in both the background and foreground generation, so that the images produced by both networks will merge into a coherent image. Each generator is a composition of sub-modules applied back to back, with intermediate pre-images  $(A_{bg}, A_{fg})$ :

$$I_{bq} = G_{bq_1}(A_{bq}), \qquad A_{bq} = G_{bq_0}(v_{bq}, z) \qquad (3)$$

$$(I_{fg}, I_m) = G_{fg_2}(G_{fg_1}(A_{fg}, v_p), v_c), \qquad A_{fg} = G_{fg_0}(v_p, z) \qquad (4)$$

The final generated image is: (where  $\circ$  denotes element-wise multiplication)

$$I = I_{bg} \circ (1 - I_m) + I_{fg} \circ I_m \tag{5}$$

**Encoders** Unlike FineGAN, which performs only the generation task, our method requires the use of encoders. We introduce three encoders: background encoder  $E_{bg}$ , shape encoder  $E_p$ , and style encoder  $E_c$ . They run in semi-parallel to predict both the latent codes  $(v_{bg}, v_p, v_c, z)$  of an input image and the underlying one-hot vectors  $(e_{bg}, e_p, e_c)$ . All encoders are fed with image I as input. The background encoder is also fed with the mask  $I_m$ . During image reconstruction, there is no initial image mask, therefore it first has to be generated by encoding the shape and style features and applying the foreground generator. The lack of ground-truth mask is why the encoders do not run fully in parallel. In addition, the background and shape encoders also produce bypass tensors  $(B_{bg}, B_{fg})$  to be used as skip connections between the encoders and the generators.

$$(B_{bg}) = E_{bg}(I, I_m) \tag{6}$$

$$(\hat{e}_p, \mu_p, \sigma_p, B_{fg}, \mu_z, \sigma_z) = E_p(I) \tag{7}$$

$$(\hat{e}_c, \mu_c, \sigma_c) = E_c(I) \tag{8}$$

Following Variational Auto-Encoder literature,  $(\mu, \sigma)$  are three paired vectors of sizes  $(d_z, d_p, d_c)$  defining the mean and variance to sample each element of  $(\hat{z}, \hat{v}_p, \hat{v}_c)$  from a gaussian distribution.

**Mixup** At the intersection between the encoders and generators, we introduce a novel method to merge information coded by the encoders and information produced by the embeddings and lower levels of the generators. The mixup module [35], mixes two input variables with a weight parameter  $\beta$ . The rationale behind this application is that during generation there is no data coming from the encoders, so the mixup is turned off and only information from the embeddings and lower levels of the generators are passed forward. During reconstruction, we want our method to utilize the skip connections to improve performance and also use the predicted embeddings  $(v_p, v_c)$  to represent the object's shape and style. The contrast between the two paths leads to a difficulty in optimizing them simultaneously. The introduction of the mixup simplifies this by having both paths active during forward path and back-propagation. In contrast to regular residual connections, the ever changing  $\beta$  used in the mixup forces both inputs to be independent representations and not complement each other. The mixup modules at the vector embeddings level (mixup1 and mixup2 in Fig. 2) mix the vectors  $(v_p, v_c)$  given by the embeddings  $(V_p, V_c)$ , Eq. 2, with the predicted vectors  $(\hat{v}_p, \hat{v}_c)$  produced by the encoders  $(E_p, E_c)$ , Eq. 7,8. The mixture of features leads to both the embeddings and the encoders being optimized for reconstructing the object. This has two benefits. First, it trains the encoders to properly code the images, which improves clustering and learns image-to-image translation implicitly. Second, it trains the embeddings to represent the real object classes, which improves the generation task.

The mixup modules at the skip connections (mixup3 and mixup4 in Fig. 2) mix the pre-image tensors  $(A_{bg}, A_{fg})$ , Eq. 3,4, with the bypass tensors  $(B_{bg}, B_{fg})$ , Eq. 6,7. It serves to create the condition where the reconstruction path will be simultaneously dependent on the bypass and on the lower stage of the generators. This way, at any time we can choose any  $\beta$  or even pass only the bypass or only the pre-image and result in an almost identical image.

Given two inputs and a parameter  $\beta$ , the mixup is defined as follows:

$$v_{p_{mix}} = v_p \circ (1 - \beta_1) + \hat{v}_p \circ \beta_1, \quad v_{c_{mix}} = v_c \circ (1 - \beta_2) + \hat{v}_c \circ \beta_2$$

$$A_{fg_{mix}} = A_{fg} \circ (1 - \beta_3) + B_{fg} \circ \beta_3, \quad A_{bg_{mix}} = A_{bg} \circ (1 - \beta_4) + B_{bg} \circ \beta_4$$
(9)

In our implementation,  $\beta_1, \beta_2 \in [0, 1]$  and  $\beta_3, \beta_4 \in [0.5, 1]$ , are sampled in each iteration for each instance in the batch. At reconstruction, the mixed features  $(v_{p_{mix}}, v_{c_{mix}}, A_{fg_{mix}}, A_{bg_{mix}})$  replace the features  $(v_p, v_c, A_{fg}, A_{bg})$  in Eq. 3,4 as input to the generators. For illustration, please refer to Fig. 2.

**GLU Layer Normalization** Following StackGANv2 and FineGAN architecture, we apply GLU [9] activation in the generators. Due to the multiple paths, the large scale and high complexity of our method, batch normalization was unstable for our low batch size, and, increasing the batch size was not an option. As a solution, we switched to layer normalization, which is not affected by the batch size. We fused the normalization and activation into a single module termed "GLU Layer Normalization". Given an input x with  $x_L, x_R$  representing an equal split in the channel axis (left/right):

$$GLU(x_L, x_R) = x_L \circ \text{Sigmoid}(x_R)$$
  
GLU-LNorm $(x_L, x_R) = GLU(\text{LNorm}(x_L), x_R)$  (10)

In this method, the normalization is only applied on  $x_L$ . The input to the sigmoid,  $x_R$ , is not normalized. This is favorable, because  $x_R$  serves as a mask on  $x_L$ , and normalizing it across the channels contradicts this goal.

**Discriminators** Following FineGAN, the two discriminators are adversarial opponents on the outputs  $I_{bg}$ , I. The background discriminator  $D_{bg}$  has two tasks, with a separate output for each. The tasks are as follows: (i) patch-wise prediction if the input image is real or fake when presented with either a real or fake background image, annotated as  $D_{bgA}$ . (ii) patch-wise prediction if the input image or not when presented with either a real background image or a real object image, annotated as  $D_{bgB}$ . The background generator is hereby optimized to generate images that look like real images and

do not contain object features. In addition, when performing the reconstruction path on fake images, we also extract a hidden layer output and apply perceptual loss between generated and reconstructed backgrounds, annotated as  $D_{bg_C}$ , to reduce the perceptual distance between the original and the reconstructed image.

The image discriminator  $D_c$  receives real images from  $X_c$  or generated fake images, and also has two tasks: (i) predict if the input image is real or fake, annotated as  $D_{c_A}$ . (ii) predict the child class  $\phi_c$  of the image, annotated as  $D_{c_B}$ , as in all InfoGAN-influenced methods. This trains the foreground generator to generate images that look real and represent the conditioned child class. In addition, we also extract a hidden layer output and apply perceptual loss between generated and reconstructed foreground images, annotated as  $D_{c_C}$ .

### 4 Training

To train to solve various tasks, we perform in each iteration two different paths through the model, by connecting the various sub-networks in a specific order.

#### 4.1 Generation path

The generation path is described in Fig. 1, Eq. 2–5. For illustrations, see Fig. 3. The inputs for this path are  $e_{bg}$ ,  $e_p$ ,  $e_c$ , z. During generation, the model learns to generate image I in a way that relies on generating a background  $I_{bg}$ , foreground  $I_{fg}$ , and mask  $I_m$  images. The discriminators are trained along with the generators and produce an adversarial training signal. In addition, the encoders are also trained to retrieve the latent variables from the generated images, as a self-supervised task.

The losses in this path can be put into four groups: adversarial losses, classification losses, distance losses, and regularizations. For brevity, e represents the dependence on all prior codes  $(e_{bg}, e_p, e_c)$ . Similarly, G(e, z) represents the full generation of the final image, Eq. 3–5.

Adversarial losses These involve the two discriminators and are derived from the minimax equation:  $\min_G \max_D \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))]$ , for a generic generator G and discriminator D. The concrete GAN loss is the sum of the losses for the separation between real/fake background, the separation between background/object and the separation between real/fake object.

For the discriminators, where  $X_{bg}$ ,  $X_c$  are the sets of real background images and real object images, the losses are:

$$\mathcal{L}_{D_{bg_{A}}} = \mathbb{E}_{x \sim X_{bg}} [\log(D_{bg_{A}}(x))] + \mathbb{E}_{e_{bg},z} [\log(1 - D_{bg_{A}}(G_{bg}(e_{bg}, z)))] \\
\mathcal{L}_{D_{bg_{B}}} = \mathbb{E}_{x \sim X_{bg}} [\log(D_{bg_{B}}(x))] + \mathbb{E}_{x \sim X_{c}} [\log(1 - D_{bg_{B}}(x))] \\
\mathcal{L}_{D_{c_{A}}} = \mathbb{E}_{x \sim X_{c}} [\log(D_{c_{A}}(x))] + \mathbb{E}_{e,z} [\log(1 - D_{c_{A}}(G(e, z)))] \\
\mathcal{L}_{D} = 10 \cdot \mathcal{L}_{D_{bg_{A}}} + \mathcal{L}_{D_{bg_{B}}} + \mathcal{L}_{D_{c_{A}}}$$
(11)

For the generators, the losses are:

$$\mathcal{L}_{G_{bg_{A}}} = \mathbb{E}_{e_{bg},z}[\log(D_{bg_{A}}(G_{bg}(e_{bg},z)))]$$

$$\mathcal{L}_{G_{bg_{B}}} = \mathbb{E}_{e_{bg},z}[\log(D_{bg_{B}}(G_{bg}(e_{bg},z)))]$$

$$\mathcal{L}_{G_{c_{A}}} = \mathbb{E}_{e,z}[\log(D_{c_{A}}(G(e,z)))]$$

$$\mathcal{L}_{G} = 10 \cdot \mathcal{L}_{G_{bg_{A}}} + \mathcal{L}_{G_{bg_{B}}} + \mathcal{L}_{G_{c_{A}}}$$
(12)

**Classification losses** These losses optimize the generators to generate distinguished images for each style and shape priors and optimize the encoders to retrieve the prior classes. We use the cross entropy loss between the conditioned classes  $(\phi_p, \phi_c)$  and the encoders' predictions  $(\hat{e}_p, \hat{e}_c)$  form Eq.7,8. In addition, we use the auxiliary task  $D_{C_B}$ .

$$\mathcal{L}_E = \operatorname{CE}(\hat{e}_p, \phi_p) + \operatorname{CE}(\hat{e}_c, \phi_c) + \operatorname{CE}(D_{c_B}(I), \phi_c)$$
(13)

**Distance losses** We train the encoders to minimize the mean squared error between the vectors in the latent space produced during generation and their predicted counterparts. These vectors are used in the reconstruction path, thus this self-supervised task assists in this regard. We minimize the distance between the pre-images and bypasses  $(A_{bg}, A_{fg}, B_{bg}, B_{fg})$ , and between the latent vectors  $(v_p, v_c)$  and the mean vectors  $\mu_p, \mu_c$  used to sample the latent code  $(\hat{v}_p, \hat{v}_c)$ .

$$\mathcal{L}_{\text{MSE}} = \text{MSE}(v_c, \mu_c) + \text{MSE}(v_p, \mu_p) + \text{MSE}(A_{fg}, B_{fg}) + \text{MSE}(A_{bg}, B_{bg}) \quad (14)$$

**Regularization losses** For regularization, a loss term is applied on the latent codes  $(v_{bg}, v_p, v_c)$ , annotated as  $\mathcal{L}_{R_v}$ , and on the foreground mask  $I_m$ , annotated as  $\mathcal{L}_{R_M}$ . They are detailed in the supplementary.

All the losses are summed together to the total loss:

$$\mathcal{L}_{\text{GEN}} = \mathcal{L}_G + \mathcal{L}_E + \mathcal{L}_{\text{MSE}} + 0.1 \cdot \mathcal{L}_{R_v} + 2 \cdot \mathcal{L}_{R_M}$$
(15)

#### 4.2 Reconstruction path

The reconstruction path is described in Fig. 2. For illustrations, see Fig. 4. The input is an image I. The precise flow is: (1) encode the foreground through the shape and style encoders  $(E_p, E_c)$ , Eq. 7,8, (2) generate a foreground image and mask with the foreground generator  $(G_{fg})$ , Eq. 4, (3) encode the image and mask through the background encoder  $(E_{bg})$ , Eq. 6, (4) generate the background image with the background generator  $(G_{bg})$ , Eq. 3, and (5) compose the final image with  $I_{fg}$ ,  $I_{bg}$ ,  $I_m$ , Eq. 5. In addition, the mixup is applied as in Eq. 9 between encoding and generation. This path optimizes the clustering and segmentation tasks directly and also implicitly optimizes the generation task by reconstruction real images. We perform the reconstruction path on both real images and generated images from the generation path. This fully utilizes the information available to learn all tasks with minimal supervision.

The losses in this path can be put in three groups: statistical losses, reconstruction losses, and perceptual losses. **Statistical losses** As in Variational Auto-Encoders [18], we compare the Kullback-Leiber Divergence between the latent variables encoded by the encoders  $(\hat{v}_p, \hat{v}_c, \hat{z})$  to a multivariate gaussian distribution. For the pose vector z, we used the standard normal distribution with covariance matrix equal to the identity matrix ( $\Sigma = I_{d_z}$ ) and a zero mean vector ( $\mu = 0$ ). For the shape and style vectors ( $\hat{v}_p, \hat{v}_c$ ) we still use identity  $\Sigma$ , but since they should match their latent code ( $v_p, v_c$ ), we use these latent codes as the target mean.

$$\mathcal{L}_{\text{VAE}_{p}} = D_{\text{KL}}(\mathcal{N}(\mu_{p}, \text{diag}(\sigma_{p})) \| \mathcal{N}(v_{p}, I_{d_{p}})), \mathcal{L}_{\text{VAE}_{c}} = D_{\text{KL}}(\mathcal{N}(\mu_{c}, \text{diag}(\sigma_{c})) \| \mathcal{N}(v_{c}, I_{d_{c}})), \mathcal{L}_{\text{VAE}_{z}} = D_{\text{KL}}(\mathcal{N}(\mu_{z}, \text{diag}(\sigma_{z})) \| \mathcal{N}(\mathbf{0}, I_{d_{z}})), \mathcal{L}_{\text{VAE}_{z}} = \mathcal{L}_{\text{VAE}_{p}} + \mathcal{L}_{\text{VAE}_{c}} + \mathcal{L}_{\text{VAE}_{z}}$$
(16)

**Reconstruction losses** The reconstruction losses are a set of L1 losses that compare the difference between the input image to the output. The network trains at reconstructing both real and fake images. For fake images, we have the extra self-supervision to also compare reconstruction of the background image and foreground mask.

$$\mathcal{L}_{\text{REC}} = \begin{cases} \text{L1}(I, \hat{I}) &, \text{real} \\ \text{L1}(I, \hat{I}) + \text{L1}(I_{bg}, \hat{I}_{bg}) + \text{L1}(I_m, \hat{I}_m) &, \text{fake} \end{cases}$$
(17)

**Perceptual losses** Comparing images to their ground-truth counterpart is known to produce blurred images; Perceptual loss [15] is known to aid in producing sharper images with more visible context [36] by comparing the images on the feature level as well. The perceptual loss is often used along with a pretrained network, but this relies on added supervision. In our case, we use the discriminators as feature extractors. We use the notation  $D_{bg_C}$ ,  $D_{c_C}$  from Sec. 3 to describe the extraction of the hidden layers used for this comparison.

$$\mathcal{L}_{\text{PER}} = \begin{cases} \text{L2}_{D_{c_{C}}}(I, \hat{I}) &, \text{real} \\ \text{L2}_{D_{c_{C}}}(I, \hat{I}) + \|D(D_{bg_{C}}) - D(\hat{I}_{bg})\|^{2} &, \text{fake} \end{cases}$$
(18)

All the losses are summed together to the total loss:

$$\mathcal{L}_{AE} = \mathcal{L}_{GEN} + \mathcal{L}_{VAE} + \mathcal{L}_{REC} + \mathcal{L}_{PER}$$
(19)

#### 4.3 Multi-phase training

In order to simplify training, instead of training both paths at once, we schedule the training process by phases. The phases are designed to train the network for a gradually increasing subset of tasks, starting from image-level tasks (generating images) to semantic tasks (semantic segmentation of the foreground, and semantic clustering) that benefit from the capabilities obtained in the generation path. In the first phase we only perform the generation path 4.1 and in the second phase we add the reconstruction path 4.2.

Without multi-phase training, the networks would be trained to generate and reconstruct images simultaneously. While the generation flow encourages a separation between the background and foreground components, the reconstruction



**Fig. 3.** Image Generation for each dataset. From top to bottom: (i) final image, (ii) foreground, (iii) foreground mask, (iv) background.



**Fig. 4.** Image reconstruction for each dataset. 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.

flow resists this separation due to the trivial solution of encoding and decoding the image in one of the paths (foreground or background) and applying an all-zero or all-one mask. In the experiments, in Tab. 1,2, we show that without multi-phase the model is incapable of learning any task.

In this controlled environment, the generators are much more likely to converge to the required setting. After a decent amount of iterations, determined in advance by a hyper-parameter, the second phase kicks in, where the model is also trained to reconstruct images, which will train the encoders on top of the generator instead of breaking it.

When entering Phase II, the fake images for both discriminators can be a result of either (i) generation path, (ii) fake image reconstruction, or (iii) real image reconstruction. We noticed that images from the reconstruction paths fail to converge to real-looking images when the discriminators were only trained by the generation path outputs. We hypothesized that this is probably due to each path producing images from a different source domain and these paths can generate very different images during training and the discriminators get overwhelmed by the different tasks and are not able to optimize them simultaneously. To solve this, upon entering Phase II, we clone each discriminator  $(D_c, D_{bg})$  twice and associate one separate clone for each path, resulting in a total of three background discriminators and another three for the foreground. In this setting, each path receives the adversarial signal that is concentrated only at improving that path.

### 5 Experiments

We train the network for 600,000 iterations, with batch size 20. All sub-networks are optimized using Adam [19], with lr=2e-4. Phase I duration is 200,000 iterations and Phase II 400,000. Within Phase II, we start with training only on fake images and real image reconstruction starts after another 200,000 iterations.



**Fig. 5.** Conditional Generation. From left to right: (i) real images, (ii-vi) generation of images with the encoded parent and child codes and a different vector z per column, (vii) FineGAN [26] + our encoders, (viii) StackGANv2 [34] + our encoders.

Fig. 6. Style Transfer. From left to right: (i) real images. (ii-vi) reconstructed images when the child code  $e_c$  is switched with a code from a selected category, (vii) FineGAN [26] + our encoders, (viii) StackGANv2 [34] + our encoders.

**Table 1.** Quantitative generation results. FID $\downarrow$ , IS $\uparrow$ , CFID $\downarrow$ , CIS $\uparrow$ 

	Birds				Dogs			Cars				
Model	FID	IS	CFID	CIS	FID	IS	CFID	CIS	FID	IS	CFID	CIS
Dataset	0	163.6	0	47.9	0	114.2	0	77.1	0	163.1	0	55.4
StackGANv2 FineGAN	$\begin{array}{c} 21.4\\ 23.0 \end{array}$	$\begin{array}{c} 67.0\\ 66.4\end{array}$	$96.8 \\ 65.3$	$\begin{array}{c} 15.0\\ 24.7\end{array}$	$56.7 \\ 54.9$	82.4 83.1	$\begin{array}{c} 184.7\\ 100.4 \end{array}$	$\begin{array}{c} 10.2\\ 15.7\end{array}$	$\begin{array}{c} 25.0\\ 24.8 \end{array}$	$\begin{array}{c} 88.1\\ 86.2 \end{array}$	$\begin{array}{c} 190.3\\ 126.0 \end{array}$	$\begin{array}{c} 13.3\\ 13.6 \end{array}$
OneGAN no real recon Phase I only no multi-phase	<b>20.5</b> 22.3 23.9 e196.5	<b>67.4</b> 65.6 63.2 11.0	<b>55.2</b> 58.6 59.1 356.1	<b>30.7</b> 25.6 21.6 2.3	<b>48.7</b> 55.4 56.1 217.8	<b>89.7</b> 84.2 82.0 16.9	<b>92.0</b> 95.3 97.8 543.2	<b>19.6</b> 17.0 16.8 1.7	<b>24.2</b> 25.1 25.4 264.7	<b>90.3</b> 88.2 87.7 23.4	<b>100.7</b> 104.3 106.1 767.9	<b>18.7</b> 15.5 13.4 3.9

We evaluate our model on various tasks against the state of the art methods. Since no other model can solve all these tasks, we evaluate against different methods in each task. Depending on availability, some baselines were pre-trained models released by the authors and some were trained from scratch with the authors' official code and instructions.

**Datasets** We evaluate our model with three datasets of fine-grained categorization. **Caltech-UCSD Birds-200-2011 (Birds)** [29]: This dataset consists of 11,788 images of 200 classes of birds, annotated with bounding boxes and segmentation masks. **Stanford Dogs (Dogs)** [20]: This dataset consists of 20,580 images of 120 classes of dogs, annotated with bounding boxes. For evaluation, target segmentation masks were generated by a pre-trained DeepLabV3 [5] model on the COCO [22] dataset. The pre-trained model was acquired from the gluoncv toolkit [12]. **Stanford Cars (Cars)** [17]: This dataset consists of 16,185 images of 196 classes of cars, annotated with bounding boxes. Segmentation masks were generated as above with the pre-trained DeepLabV3 model.

Similarly to FineGAN, before training the model, we produced a background subset by cutting background patches with the bounding boxes. In addition to

		ç	Segme	entatio	n				Clust	ering		
	Bi	irds	D	ogs	С	ars	Bi	rds	D	ogs	Ca	ars
Model	IOU	DICE	IOU	DICE	IOU	DICE	ACC	NMI	ACC	NMI	ACC	NMI
ReDO	46.5	60.2	38.4	52.8	16.2	26.2	X	X	X	X	X	X
WNet	24.8	38.9	47.7	62.1	52.8	67.6	X	X	X	X	X	X
UISB <sup>§</sup>	44.2	60.1	62.7	75.5	64.7	77.5	X	X	X	X	X	X
IIC-seg stf-3 <sup>§</sup>	36.5	50.2	58.5	71.5	58.5	71.5	X	X	X	X	X	X
$IIC-seg stf^{\S}$	35.2	50.4	56.6	70.2	58.8	71.7	X	X	X	X	X	X
$JULE^{\dagger}$	X	X	X	X	X	X	.045	.204	.043	.142	.046	.232
$DEPICT^{\dagger}$	X	×	X	×	X	X	.061	.290	.052	.182	.063	.329
IIC-cluster	X	×	X	X	X	X	.084	.345	.060	.200	.056	.254
StackGANv2	X	×	X	X	X	X	$.057^{*}$	$.253^{*}$	$.040^{*}$	$.139^{*}$	$.039^{*}$	$.174^{*}$
FineGAN	$44.5^{*}$	$56.9^{*}$	$48.7^{*}$	$59.3^{*}$	$53.2^{*}$	$60.3^{*}$	$.086^{*}$	$.349^{*}$	$.059^{*}$	$.194^{*}$	$.051^{*}$	.233*
OneGAN	55.5	69.2	71.0	81.7	71.2	82.6	.101	.391	.073	.211	.060	.272
no real recon	53.5	67.7	67.1	78.6	69.8	81.1	.095	.389	.062	.194	.057	.250
Phase I only	45.7	60.6	65.1	77.3	64.8	75.9	.084	.352	.058	.175	.052	.244
no multi-phase	28.2	43.2	7.4	13.6	45.9	60.5	.050	.216	.019	.082	.041	.208

**Table 2.** Segmentation and clustering results. <sup>§</sup>unfair upper bound results, obtained by selecting the best result out of many. <sup>†</sup>provided by [26]. \*model performed task by using our encoders.  $\bigstar$  model cannot perform task. Higher is better in all scores.

**Table 3.** Ablation studies on CUB: (a) normalization methods, (b) modules' behaviour, and (c) losses. Measuring FID and C-IS for generation and IOU for segmentation.

Model	FID	C-IS	IOU	Model	FID	C-IS	IOU	Model	FID	C-IS	IOU
OneGAN GLU-INorm LNorm INorm	20.5 122.0 87.5 103.4	30.7 10.2 14.5 9.8	55.5 31.3 45.4 30.1	OneGAN no bypass no mixup(1,2) no mixup(3,4)	20.5 21.2 22.6 20.9	30.7 22.8 17.5 22.2	55.5 53.3 54.1 53.8	$\begin{array}{c} \text{OneGAN} \\ \text{no loss } \mathcal{L}_{R_M} \\ \text{no loss } \mathcal{L}_{\text{VAE}} \\ \text{no loss } \mathcal{L}_{\text{PER}} \end{array}$	20.5 97.2 44.1 25.5	30.7 19.5 18.5 24.1	55.5 35.3 39.6 53.0
	(a)			(	b)				(c)		

FineGAN, the bounding boxes were not used in any other way to train our method and we made sure that no image was used for both foreground and background examples. This was done by splitting the dataset in a 80/20 ratio, and use the larger subset as foreground  $X_c$  and only the smaller subset for background  $X_{bq}$ .

Due to the different size of classes in each dataset, there is also a different size of child and parent classes in the design for each dataset. Birds:  $N_C = 200, N_P = 20$ , Dogs:  $N_C = 120, N_P = 12$ , Cars:  $N_C = 196, N_P = 14$ .

**Image generation** We compare our image generation results to FineGAN [26] and StackGANv2 [34], by relying on an InceptionV3 fine-tuned on each dataset. We evaluate our method in both IS [25] and FID [13]. In addition, we measure the conditional variants of these metrics (CIS, CFID), as presented in [2]. The conditional metrics measure the similarity between real and fake images within each class, which cannot be measured by the unconditional metrics.

Our results, reported in Tab. 1 show that OneGAN outperforms in both conditional and unconditional image generation metrics. In unconditional generation, our method and the baselines performed roughly the same, since the generators are very similar. In conditional generation, our method improves on the baseline by a large margin. StackGANv2 was the worst performing model, followed by FineGAN. This suggest that the mask-based generation, that Fine-GAN and our method rely on, generates a stronger conditioning on the object in the image. In addition, our multi-path training method improves conditional generation further, as is shown in the ablation tests. For illustration of conditional generation, see Fig. 5.

Unsupervised foreground segmentation We compare our mask prediction from the reconstruction path to the real foreground mask. We evaluate according to IOU and DICE scores. We compare against three baselines, ReDO [6]. WNet [32] and UISB [16] which are trained for each dataset separately, and a third one, IIC-seg [14], which was trained on coco-stuff and coco-stuff-3 (a subset). While coco-stuff is a different dataset than the ones we used, it contains all the relevant classes. ReDO and WNet produce a foreground mask which we compare to the ground-truth similarly to how we evaluate our model. UISB is an iterative method that produces a final segmentation with a varying number of classes between 2 and 100. We iterated UISB on each image 50 times. The output was usually between 4-20 classes. Since there is no labeling of the foreground or background classes, this method cannot be immediately used for this task. In order to get an evaluation, we look for each image for the class that has the highest IOU with the ground-truth foreground. The rest of the classes are merged to a single background class. We then repeat with a single background class and the rest merged into foreground. Finally, taking the best out of the two options, each obtained by using an oracle to select out of many options, which provides a liberal upper bound on the performance of UISB. IIC also produces a multi-class segmentation map, we use it in the same way we use UISB by taking the best class for either background or foreground in respect to IOU. IIC has 2-headed output, one for the main task and one for over-clustering. For coco-stuff trained IIC, we look for the best mask in one of the 15 classes of the main head. For cocostuff-3 trained IIC, the main head is trained to cluster sky/ground/plants, so we look for the best mask in one of the 15 classes of the over-clustering head. Fine-GAN cannot perform segmentation, since it does not have a reconstruction path. But we added an additional baseline by training FineGAN with our encoders to allow such path. The results in Table. 2 show that our method outperforms all the baselines. The ablation show that the biggest contribution comes from the reconstruction path and the multi-phase scheduling.

**Unsupervised clustering** We compare our method against JULE [33], DE-PICT [10] and IIC-cluster [14]. In addition, we added the baselines of Stack-GANv2 and FineGAN trained with our encoders. In this task, we evaluate how well the encoders are capable of clustering real images. The results show that OneGAN outperforms the other methods for both Birds and Dogs datasets. For Cars, our model was second after DEPICT. By looking at the generated images, this can be explained by the fact our method clusters the cars based more on color and less on car model. This aligns with the conditional generation scores, where the scores for Cars were lower than for the other datasets.

**Image to image translation** To further evaluate our model, we show its capability to transfer an input image to a target category. The results can be seen in Fig. 6. Even though our model was never trained on this task, the disentanglement between the shape and the texture enables this translation simply by passing a different child code during reconstruction. By selecting different child codes, we can manipulate the appearance of the object to any of the child categories. In contrast, FineGAN and StackGANv2 are unable to perform this task correctly as there is no learned disentanglement in StackGANv2's case and no bypass connection in FineGAN's case to allow good reconstruction.

**Object removal and inpainting** Through the reconstruction task, our model is also capable of performing automatic object removal and background reconstruction, see Fig. 4. In contrast to other known method for inpainting, due to the lack of perfect ground-truth mask, our model does not only fill the missing pixels but fully reconstructs the background image. As a result, the background image is not identical to the original background, but it is semantically similar to it. we compare our method with previous work in the supplementary.

Ablation study In Tab. 1,2, we provide multiple versions of our method for ablation. In the version without real reconstruction, we only add fake image reconstruction in Phase II, meaning that real images did not pass through the network during training. Another variant employs only the first phase of training. Finally, a third variant trains without multi-phase scheduling. These tests show the contribution of the multiple paths and the multi-phase scheduling. In Tab. 3, we provide an extensive ablation study on three aspects. In (a), we compared layer and instance normalization [28] methods in the generators. Our "GLU layer normalization" outperformed all other options. In (b), we turned of intersection modules between encoders and generators. The experiment shows that these models strongly improve the CIS, which explains why our method outperformed FineGAN and StackGANv2 in conditional generation. In (c), we evaluated the contribution of selected novel losses, which affected all scores. Together, all these experiments show the contribution of the proposed novelties in our method.

# 6 Conclusions

By building a single model to handle multiple unsupervised tasks at once, we convincingly demonstrate the power of co-training, by surpassing the performance of the best in class methods for each task. This capability is enabled by a complex architecture with many sub-networks. Considering biological visual system, one can expect future architectures to be complex and to contain multiple pathways between the various modules. However, supporting this complexity during training is challenging. We introduce a mixup module that integrates multiple pathways in a homogenized manner and a multi-phase training, which helps to avoid some tasks dominating over the others.

# Acknowledgement

This project has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC CoG 725974).

# References

- 1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
- Benny, Y., Galanti, T., Benaim, S., Wolf, L.: Evaluation metrics for conditional image generation. arXiv preprint arXiv:2004.12361 (2020)
- Bielski, A., Favaro, P.: Emergence of object segmentation in perturbed generative models. In: Advances in Neural Information Processing Systems. pp. 7256–7266 (2019)
- Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence 40(4), 834–848 (2017)
- 5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
- Chen, M., Artières, T., Denoyer, L.: Unsupervised object segmentation by redrawing. arXiv preprint arXiv:1905.13539 (2019)
- Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Advances in neural information processing systems. pp. 2172–2180 (2016)
- Croitoru, I., Bogolin, S.V., Leordeanu, M.: Unsupervised learning of foreground object detection. arXiv preprint arXiv:1808.04593 (2018)
- Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 933–941. JMLR. org (2017)
- Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W., Huang, H.: Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5736–5745 (2017)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
- Guo, J., He, H., He, T., Lausen, L., Li, M., Lin, H., Shi, X., Wang, C., Xie, J., Zha, S., Zhang, A., Zhang, H., Zhang, Z., Zhang, Z., Zheng, S.: Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. arXiv preprint arXiv:1907.04433 (2019)
- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. pp. 6626–6637 (2017)
- Ji, X., Henriques, J.F., Vedaldi, A.: Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9865–9874 (2019)

- Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. pp. 694–711. Springer (2016)
- Kanezaki, A.: Unsupervised image segmentation by backpropagation. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 1543–1547. IEEE (2018)
- Khosla, A., Jayadevaprakash, N., Yao, B., Fei-Fei, L.: Novel dataset for fine-grained image categorization. In: First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition. Colorado Springs, CO (June 2011)
- 18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. stat 1050, 1 (2014)
- 19. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: The International Conference on Learning Representations (ICLR) (2016)
- Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for finegrained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia (2013)
- 21. Larsen, A.B.L., Sønderby, S.K., Larochelle, H., Winther, O.: Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300 (2015)
- Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
- Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
- Ronneberger, O., P.Fischer, Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI). LNCS, vol. 9351, pp. 234-241. Springer (2015), http: //lmb.informatik.uni-freiburg.de//Publications/2015/RFB15a, (available on arXiv:1505.04597 [cs.CV])
- Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. arXiv preprint arXiv:1606.03498 (2016)
- Singh, K.K., Ojha, U., Lee, Y.J.: Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. arXiv preprint arXiv:1811.11155 (2018)
- Sultana, M., Mahmood, A., Javed, S., Jung, S.K.: Unsupervised deep context prediction for background estimation and foreground segmentation. Machine Vision and Applications 30(3), 375–395 (2019)
- Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
- Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)
- Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. arXiv preprint arXiv:1908.07919 (2019)
- Wang, Y., Choi, J., Chen, Y., Li, S., Huang, Q., Zhang, K., Lee, M.S., Kuo, C.C.J.: Unsupervised video object segmentation with distractor-aware online adaptation. arXiv preprint arXiv:1812.07712 (2018)
- Xia, X., Kulis, B.: W-net: A deep model for fully unsupervised image segmentation. arXiv preprint arXiv:1711.08506 (2017)

- 33. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5147–5156 (2016)
- 34. Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., Metaxas, D.N.: Stackgan++: Realistic image synthesis with stacked generative adversarial networks. IEEE transactions on pattern analysis and machine intelligence 41(8), 1947–1962 (2018)
- Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
- 36. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–595 (2018)

# A Summary of notation

For convenience, in Tab. 4 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.

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

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. 6–12, with some modules that are frequently used listed in Tab. 5. 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. 7, 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. 8, 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. 9, 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 sidetask 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. 10. It can be observed that DIP works relatively well when using a perfectly covering mask, but fails when the mask is not perfect 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.

	Symbol	Description	Computed as (or a comments)
Variables	$\begin{array}{l} \phi_{c} \in [1, N_{C}] \\ \phi_{p} \in [1, N_{P}] \\ e_{c} \in \{0, 1\}^{N_{C}} \\ e_{p} \in \{0, 1\}^{N_{P}} \\ e_{bg} \in \{0, 1\}^{N_{P}} \\ z \in \mathbb{R}^{d_{z}} \\ v_{c} \in \mathbb{R}^{d_{c}} \\ v_{p} \in \mathbb{R}^{d_{p}} \\ v_{bg} \in \mathbb{R}^{d_{bg}} \\ A_{fg} \\ A_{bg} \\ I_{m} \\ I_{fg} \\ I_{bg} \\ I \\ B_{fg} \\ B_{bg} \\ X_{c} \end{array}$	child class parent class child class one-hot vector (style) parent class one-hot vector (shape) background one-hot vector pose code style code vector shape code vector background code vector foreground pre-image background pre-image foreground mask foreground image background image full image foreground bypass background bypass image domain	$\begin{split} e_{c}[i] &= \delta_{i,\phi_{c}} \\ e_{p}[i] &= \delta_{i,\phi_{p}} \\ e_{bg} &= e_{p} \\ z[i] &\sim \mathcal{N}(0,1) \\ v_{c} &= V_{c_{0}}(e_{c}) \\ v_{p} &= V_{p_{0}}(e_{p}) \\ v_{bg} &= V_{bg_{0}}(e_{bg}) \\ A_{fg} &= G_{fg_{0}}(v_{p},z) \\ A_{bg} &= G_{bg_{0}}(v_{bg},z) \\ (I_{fg}, I_{m}) &= G_{fg_{2}}(G_{fg_{1}}(A_{fg},v_{p}),v_{c}) \\ I_{bg} &= G_{bg_{1}}(A_{bg}) \\ I &= I_{bg} \circ (1 - I_{m}) + I_{fg} \circ I_{m} \\ B_{fg} &= E_{p_{1}}(I) \\ B_{bg} &= E_{bg_{1}}(I, I_{m}) \end{split}$
Networks	$\begin{array}{c} X_{bg} \\ \hline V_c \\ V_p \\ V_{bg} \\ G_{fg} \\ G_{bg} \\ E_c \\ E_p \\ E_{bg} \\ D_c \\ D_{bg} \end{array}$	background image domain embedding LUT of child class embedding LUT of parent class embedding LUT of background foreground generator background generator style encoder content encoder image discriminator background discriminator	$G_{fg_{2}}(G_{fg_{1}}(G_{fg_{0}}(v_{p}, z), v_{p}), v_{c})$ $G_{bg_{1}}(G_{bg_{0}}(v_{bg}, z))$ $E_{c}(I)$ $E_{p}(I)$ $E_{bg}(I, I_{m})$
Parameters	$N_C$ $N_P$ $d_z$ $d_c$ $d_p$ $d_{bg}$ $H, W$	number of child classes number of parent classes dimensionality of pose code dimensionality of style code dimensionality of shape code dimensionality of background code size of image	Depends on the dataset. $N_P < N_C$ , Depends on the dataset. $d_z = 100$ $d_c = 32$ $d_p = 16$ $d_{bg} = 32$ H = W = 128

 Table 4. The components of the OneGAN model



Fig. 7. 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. 8.** 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. 9. 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. 10.** 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.

Module	layers	input	output
	ChannelSplit	x	$x_L, x_R$
CLUINO	LayerNorm	$x_L$	$x'_L$
GLU-LN0III	Sigmoid	$x_R$	$x'_R$
	Multiply	$x_L^\prime, x_R^\prime$	-
	Upsample2d $(S/2, S)$	-	-
UPBlk	$K3P1Conv2d(c_i, 2c_o)$	-	-
$(c_i, c_o, S)$	GLU-LNorm	-	-
	$K4S2P1Conv2d(c_i, c_o)$	-	-
DOWNBlk	LayerNorm	-	-
$(c_i, c_o)$	lReLU(0.2)	-	-
	$K3P1Conv2d(c_i, 2c_i)$	x	-
	GLU-LNorm	-	-
RESBlk0	$K3P1Conv2d(c_i, 2c_i)$	-	-
$(c_i)$	GLU-LNorm	-	d
	Add	x, d	-
	$K3P1Conv2d(c_i + d, 2c_i)$	-	-
	GLU-LNorm	-	-
DECDI	$\text{RESBlk0}(c_i)$	-	-
RESBIR	$RESBlk0(c_i)$	-	-
$(c_i, a, c_o)$	$K3P1Conv2d(c_i, 2c_o)$	-	-
	GLU-LNorm	-	-

 Table 5. General modules

**Table 6.** Background Generator  $G_{bg}$ 

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

Module	e layers	input	output
$V_p$	$\operatorname{Linear}(N_P, d_p)$	$e_p$	$v_p$
$V_c$	$Linear(N_C, d_c)$	$e_c$	$v_c$
$G_{fg_0}$	Linear $(d_p + d_z, 32768)$ Reshape(2048,4,4) GLU-LNorm UPBlk(1024,512,8) UPBlk(512,256,16)	v <sub>p</sub> , z	$A_{fg}$
$G_{fg_1}$	$UPBlk(256,128,32) \\ UPBlk(128,64,64) \\ UPBlk(64,3,128)$	$A_{fg}$	- $C_{fg_0}$
$G_{fg_2}$	$\begin{array}{c} \text{RESBlk}(64, d_p, 32) \\ \text{RESBlk}(32, d_c, 16) \end{array}$	$\begin{array}{c} C_{fg_0}, v_p \\ C_{fg_1}, v_c \end{array}$	$\begin{array}{c} C_{fg_1} \\ C_{fg_2} \end{array}$
	$\frac{\text{K3P1Conv2d}(16,3) + \text{tanh}}{\text{K3P1Conv2d}(16,1) + \text{sigmoid}}$	$C_{fg_2}$ $C_{fg_2}$	$\frac{I_{fg}}{I_m}$

**Table 7.** Foreground Generator  $G_{fg}$ 

Table 8. Style Encoder  $E_c$ 

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

Module	layers	input	output
	K4S2P1Conv2d(3, 64)	Ι	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
$\mathbf{\Gamma}$	DOWNBlk(64, 128)	-	-
$L_{p_1}$	DOWNBlk(128, 256)	-	$H_p$
	K3P1Conv2d((256, 512)	$H_p$	-
	GLU-LNorm	-	-
	UPBlk(256,256)	-	$B_{fg}$
	DOWNBlk(256, 512)	$H_p$	-
	DOWNBlk(512, 1024)	-	-
	K3P1Conv2d(1024, 1024)	-	-
	LayerNorm	-	-
	lReLU(0.2)	-	-
	Reshape(16384)	-	h
	Linear(16384, 512)	h	-
F	LayerNorm	-	-
$E_{p_0}$	lReLU(0.2)	-	$h_p$
	$Linear((512, N_C))$	$h_p$	$\hat{e}_p$
	$Linear(512, d_c)$	$h_p$	$\mu_p$
	$Linear(512, d_c)$	$h_p$	$\sigma_p$
	Linear(16384, 512)	h	-
	LayerNorm	-	-
	lReLU(0.2)	-	$h_z$
	$Linear(512, d_z)$	$h_z$	$\mu_z$
	$Linear(512, d_z)$	$h_z$	$\sigma_z$

**Table 9.** Shape Encoder  $E_p$ 

Table 10. Background Encoder  $E_{bg}$ 

Modu	le layers	input	output
	K4S2P1Conv2d(4, 64)	$I, I_m$	-
	DOWNBlk(64, 128)	-	-
$\mathbf{F}$	DOWNBlk(128, 256)	-	$H_{bg}$
$L_{bg_1}$	K3P1Conv2d((256, 512))	$H_{bg}$	-
	GLU-LNorm	-	-
	UPBlk(256, 256)	-	$B_{bg}$

Module	layers	input	output
D	DownSample2d(128, 126)	Ι	-
	K4S2P0Conv2d(3, 64)	-	-
	lReLU(0.2)	-	-
	K4S2P0Conv2d(64, 128)	-	-
$D_{bg}$	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 11.** Background Discriminator  $D_{bg}$ 

Table 12. Object Discriminator  $D_c$ 

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