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

Yaniv Benny(✉) and Lior Wolf

Tel-Aviv University, Tel Aviv, Israel
yanivbenny@mail.tau.ac.il

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.

1 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,

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58574-7_31) contains supplementary material, which is available to authorized users.

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 single-handedly 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 [3,6,8,14,16,27,31,32] 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 and training techniques. 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, our method trains multiple neural networks side-by-side. Each task is solved by applying the networks in a specific order. Similarly, the model is trained by applying the networks in two different paths, with a specific set of losses.

Architecture. The compound network consists of two generators, three encoders and two discriminators. Figure 1,2 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.

Generators. The generation is performed by merging the results of two separate 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 ϕ_c and a parent class ϕ_p shared by multiple child classes. These classes are represented by the one-hot vectors (e_c, e_p).

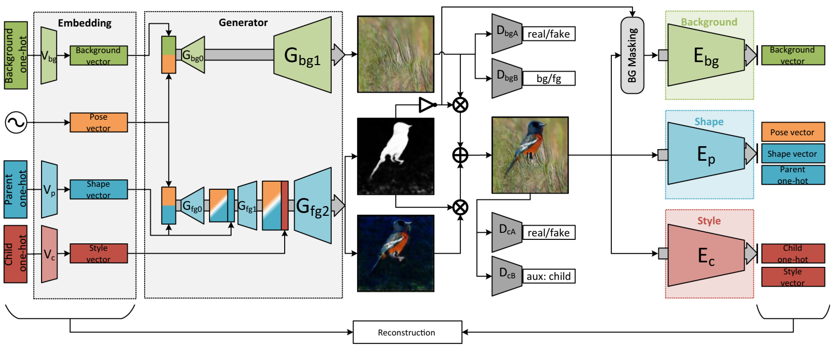

Fig. 1. Flow of the generation path. The generators decode the four priors (e_{bg}, e_p, e_c, z) and produce three separate images (foreground, background, mask) that are combined into the final image. The generated image is then coded by three encoders to retrieve the latent variables and priors.

An additional background one-hot vector e_{bg} 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.

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

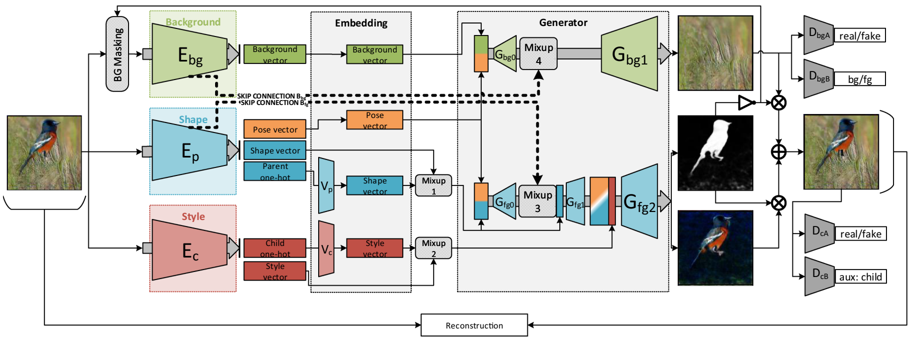

Fig. 2. Flow of the reconstruction path. The same sub-networks are rearranged to perform image reconstruction. The image is coded with the shape and style encoders and then decoded by the foreground generator to produce the foreground image and mask. Then the background encoder and generator code the 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.

The generation starts by converting the one-hot vectors into code vectors using learned embeddings. Such an embedding is often used when working with categorical values. A fourth vector z is sampled from a multi-variate gaussian distribution to represent non-categorical features.

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

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}):

$$I_{bg} = G_{bg_1}(A_{bg}), \qquad A_{bg} = G_{bg_0}(v_{bg}, z) \tag{3}$$

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

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.

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

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

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

where (μ, σ) 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 β. 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 β 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 β or even pass only the bypass or only the pre-image and result in an almost identical image.

Given two inputs and a parameter β, the mixup is defined as follows:

$$\begin{aligned} v_{p_{mix}} &= v_p \circ (1-\beta_1) + \hat{v}_p \circ \beta_1, \quad & v_{c_{mix}} &= v_c \circ (1-\beta_2) + \hat{v}_c \circ \beta_2 \\ A_{fg_{mix}} &= A_{fg} \circ (1-\beta_3) + B_{fg} \circ \beta_3, \quad & A_{bg_{mix}} &= A_{bg} \circ (1-\beta_4) + B_{bg} \circ \beta_4 \end{aligned} \tag{9}$$

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): $\text{GLU}(x_L, x_R) = x_L \circ \text{Sigmoid}(x_R)$, $\text{GLU-LNorm}(x_L, x_R) = \text{GLU}(\text{LNorm}(x_L), x_R)$. 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_{bg_A}. (ii) patch-wise prediction if the input image is a background image or not when presented with either a real background image or a real object image, annotated as D_{bg_B}. 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 ϕ_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:

$$\begin{aligned}
\mathcal{L}_{D_{bg_A}} =& \mathbb{E}_{x \sim X_{bg}}[\log(D_{bg_A}(x))] + \mathbb{E}_{e_{bg},z}[\log(1 - D_{bg_A}(G_{bg}(e_{bg}, z)))] \\
\mathcal{L}_{D_{bg_B}} =& \mathbb{E}_{x \sim X_{bg}}[\log(D_{bg_B}(x))] + \mathbb{E}_{x \sim X_c}[\log(1 - D_{bg_B}(x))] \\
\mathcal{L}_{D_{c_A}} =& \mathbb{E}_{x \sim X_c}[\log(D_{c_A}(x))] + \mathbb{E}_{e,z}[\log(1 - D_{c_A}(G(e, z)))] \\
\mathcal{L}_D =& 10 \cdot \mathcal{L}_{D_{bg_A}} + \mathcal{L}_{D_{bg_B}} + \mathcal{L}_{D_{c_A}}
\end{aligned} \tag{10}$$

For the generators, the losses are:

$$\begin{aligned}
\mathcal{L}_{G_{bg_A}} =& \mathbb{E}_{e_{bg},z}[\log(D_{bg_A}(G_{bg}(e_{bg}, z)))], \mathcal{L}_{G_{bg_B}} = \mathbb{E}_{e_{bg},z}[\log(D_{bg_B}(G_{bg}(e_{bg}, z)))] \\
\mathcal{L}_{G_{c_A}} =& \mathbb{E}_{e,z}[\log(D_{c_A}(G(e, z)))], \mathcal{L}_G = 10 \cdot \mathcal{L}_{G_{bg_A}} + \mathcal{L}_{G_{bg_B}} + \mathcal{L}_{G_{c_A}}
\end{aligned} \tag{11}$$

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 (ϕ_p, ϕ_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}.

$$\mathcal{L}_E = \text{CE}(\hat{e}_p, \phi_p) + \text{CE}(\hat{e}_c, \phi_c) + \text{CE}(D_{c_B}(I), \phi_c) \tag{12}$$

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 μ_p, μ_c used to sample the latent code $(\hat{v}_p, \hat{v}_c)$.

$$\mathcal{L}_{\text{MSE}} = \text{MSE}(v_c, \mu_c) + \text{MSE}(v_p, \mu_p) + \text{MSE}(A_{fg}, B_{fg}) + \text{MSE}(A_{bg}, B_{bg}) \tag{13}$$

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:

$$\mathcal{L}_{\text{GEN}} = \mathcal{L}_G + \mathcal{L}_E + \mathcal{L}_{\text{MSE}} + 0.1 \cdot \mathcal{L}_{R_v} + 2 \cdot \mathcal{L}_{R_M} \tag{14}$$

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 ($\mu = \mathbf{0}$). For the shape and style vectors $(\hat{v}_p, \hat{v}_c)$ we still use identity Σ, but since they should match their latent code (v_p, v_c), we use these latent codes as the target mean.

$$\begin{aligned}
\mathcal{L}_{\text{VAE}_p} &= D_{\text{KL}}(\mathcal{N}(\mu_p, \text{diag}(\sigma_p)) \| \mathcal{N}(v_p, I_{d_p})) \\
\mathcal{L}_{\text{VAE}_c} &= D_{\text{KL}}(\mathcal{N}(\mu_c, \text{diag}(\sigma_c)) \| \mathcal{N}(v_c, I_{d_c})) \\
\mathcal{L}_{\text{VAE}_z} &= D_{\text{KL}}(\mathcal{N}(\mu_z, \text{diag}(\sigma_z)) \| \mathcal{N}(\mathbf{0}, \quad I_{d_z})) \\
\mathcal{L}_{\text{VAE}} &= \mathcal{L}_{\text{VAE}_p} + \mathcal{L}_{\text{VAE}_c} + \mathcal{L}_{\text{VAE}_z}
\end{aligned} \tag{15}$$

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.

$$\mathcal{L}_{\text{REC}} = \begin{cases} \text{L1}(I, \hat{I}) & \text{, real} \\ \text{L1}(I, \hat{I}) + \text{L1}(I_{bg}, \hat{I}_{bg}) + \text{L1}(I_m, \hat{I}_m) & \text{, fake} \end{cases} \tag{16}$$

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 Sect. 3 to describe the extraction of the hidden layers used for this comparison.

$$\mathcal{L}_{\text{PER}} = \begin{cases} \text{L2}_{D_{c_C}}(I, \hat{I}) & \text{, real} \\ \text{L2}_{D_{c_C}}(I, \hat{I}) + \|D(D_{bg_C}) - D(\hat{I}_{bg})\|^2 & \text{, fake} \end{cases} \tag{17}$$

All the losses are summed together to the total loss:

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{GEN}} + \mathcal{L}_{\text{VAE}} + \mathcal{L}_{\text{REC}} + \mathcal{L}_{\text{PER}} \tag{18}$$

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 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 Table 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

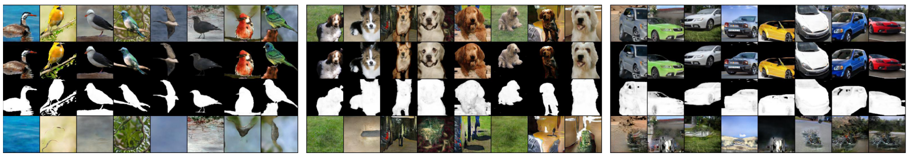

Fig. 3. Image Generation for each dataset. From top to bottom: (i) final image, (ii) foreground, (iii) foreground mask, (iv) background.

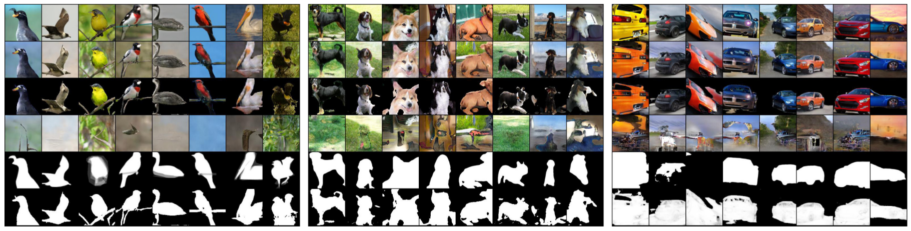

Fig. 4. Image reconstruction for each dataset. From top to bottom: (i) real image, (ii) reconstructed image, (iii) reconstructed foreground, (iv) reconstructed background, (v) ground-truth foreground mask, (vi) predicted foreground mask.

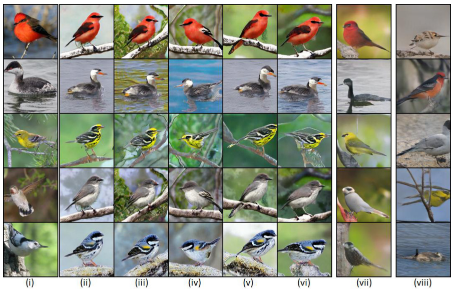

Fig. 5. Conditional Generation. From left to right: (i) real images, (ii-vi) generation of images with the encoded parent and child codes and a different vector z per column, (vii) FineGAN [26] + our encoders, (viii) StackGANv2 [34] + our encoders.

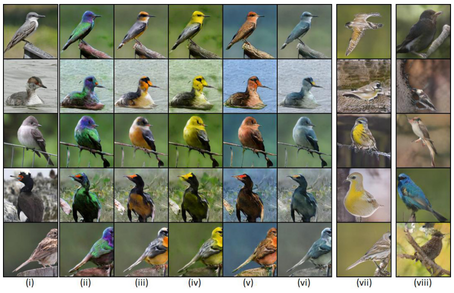

Fig. 6. Style Transfer. From left to right: (i) real images. (ii-vi) reconstructed images when the child code e_c is switched with a code from a selected category, (vii) FineGAN [26] + our encoders, (viii) StackGANv2 [34] + our encoders.

Table 1. Quantitative generation results. FID↓, IS↑, CFID↓, CIS↑

	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	21.4	67.0	96.8	15.0	56.7	82.4	184.7	10.2	25.0	88.1	190.3	13.3
FineGAN	23.0	66.4	65.3	24.7	54.9	83.1	100.4	15.7	24.8	86.2	126.0	13.6
OneGAN	**20.5**	**67.4**	**55.2**	**30.7**	**48.7**	**89.7**	**92.0**	**19.6**	**24.2**	**90.3**	**100.7**	**18.7**
No real recon	22.3	65.6	58.6	25.6	55.4	84.2	95.3	17.0	25.1	88.2	104.3	15.5
Phase I only	23.9	63.2	59.1	21.6	56.1	82.0	97.8	16.8	25.4	87.7	106.1	13.4
No multi-phase	196.5	11.0	356.1	2.3	217.8	16.9	543.2	1.7	264.7	23.4	767.9	3.9

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

	Segmentation						Clustering					
	Birds		Dogs		Cars		Birds		Dogs		Cars	
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	✗	✗	✗	✗	✗	✗
WNet	24.8	38.9	47.7	62.1	52.8	67.6	✗	✗	✗	✗	✗	✗
UISB§	44.2	60.1	62.7	75.5	64.7	77.5	✗	✗	✗	✗	✗	✗
IIC-seg stf-3§	36.5	50.2	58.5	71.5	58.5	71.5	✗	✗	✗	✗	✗	✗
IIC-seg stf§	35.2	50.4	56.6	70.2	58.8	71.7	✗	✗	✗	✗	✗	✗
JULE†	✗	✗	✗	✗	✗	✗	.045	.204	.043	.142	.046	.232
DEPICT†	✗	✗	✗	✗	✗	✗	.061	.290	.052	.182	**.063**	**.329**
IIC-cluster	✗	✗	✗	✗	✗	✗	.084	.345	.060	.200	.056	.254
StackGANv2	✗	✗	✗	✗	✗	✗	.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 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	20.5	30.7	55.5	OneGAN	20.5	30.7	55.5	OneGAN	20.5	30.7	55.5
GLU-INorm	122.0	10.2	31.3	No bypass	21.2	22.8	53.3	No loss $\mathcal{L}_{R_M}$	97.2	19.5	35.3
LNorm	87.5	14.5	45.4	No mixup(1,2)	22.6	17.5	54.1	No loss $\mathcal{L}_{\text{VAE}}$	44.1	18.5	39.6
INorm	103.4	9.8	30.1	No mixup(3,4)	20.9	22.2	53.8	No loss $\mathcal{L}_{\text{PER}}$	25.5	24.1	53.0
	(a)				(b)				(c)		

We train the network for 600,000 iterations, with batch size 20. All subnetworks 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.

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 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_{bg}.

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 Table 1 show that OneGAN outperforms in both conditional and unconditional image generation. 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. This suggest that the mask-based generation, that FineGAN 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 coco-stuff-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. FineGAN 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], DEPICT [10] and IIC-cluster [14]. In addition, we added the baselines of StackGANv2 and FineGAN trained with our encoders. We evaluate how well the encoders cluster the real images. OneGAN outperforms the other methods for both Birds and Dogs. 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 lower conditional generation score for Cars 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. In contrast, FineGAN and StackGANv2 are unable to perform this task correctly as there is no learned disentanglement in StackGANv2 and no bypass connection in FineGAN.

Object Removal and Inpainting. Our model is also capable of performing automatic object removal and background reconstruction, see Fig. 4. 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 Table 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 Table 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. 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)
2. Benny, Y., Galanti, T., Benaim, S., Wolf, L.: Evaluation metrics for conditional image generation. arXiv preprint arXiv:2004.12361 (2020)
3. Bielski, A., Favaro, P.: Emergence of object segmentation in perturbed generative models. In: Advances in Neural Information Processing Systems, pp. 7256–7266 (2019)
4. 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 Trans. Pattern Anal. Mach. Intell. **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)
6. Chen, M., Artières, T., Denoyer, L.: Unsupervised object segmentation by redrawing. arXiv preprint arXiv:1905.13539 (2019)
7. 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)
8. Croitoru, I., Bogolin, S.V., Leordeanu, M.: Unsupervised learning of foreground object detection. arXiv preprint arXiv:1808.04593 (2018)
9. 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)
10. 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)

11. 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)
12. Guo, J., et al.: Gluoncv and gluonnlp: Deep learning in computer vision and natural language processing. arXiv preprint arXiv:1907.04433 (2019)
13. 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)
14. 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)
15. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 694–711. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_43
16. Kanezaki, A.: Unsupervised image segmentation by backpropagation. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1543–1547. IEEE (2018)
17. 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)
20. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained 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)
22. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
23. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
24. 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])
25. Salimans, T., Goodfellow, I.J., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. arXiv preprint arXiv:1606.03498 (2016)
26. 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)
27. Sultana, M., Mahmood, A., Javed, S., Jung, S.K.: Unsupervised deep context prediction for background estimation and foreground segmentation. Mach. Vis. Appl. **30**(3), 375–395 (2019)
28. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)

29. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology (2011)
30. Wang, J., et al.: Deep high-resolution representation learning for visual recognition. arXiv preprint arXiv:1908.07919 (2019)
31. Wang, Y., et al.: Unsupervised video object segmentation with distractor-aware online adaptation. arXiv preprint arXiv:1812.07712 (2018)
32. 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.: Stack-gan++: realistic image synthesis with stacked generative adversarial networks. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1947–1962 (2018)
35. 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)