2017 IEEE Conference on Computer Vision and Pattern Recognition

Generative Attribute Controller with Conditional Filtered Generative Adversarial Networks

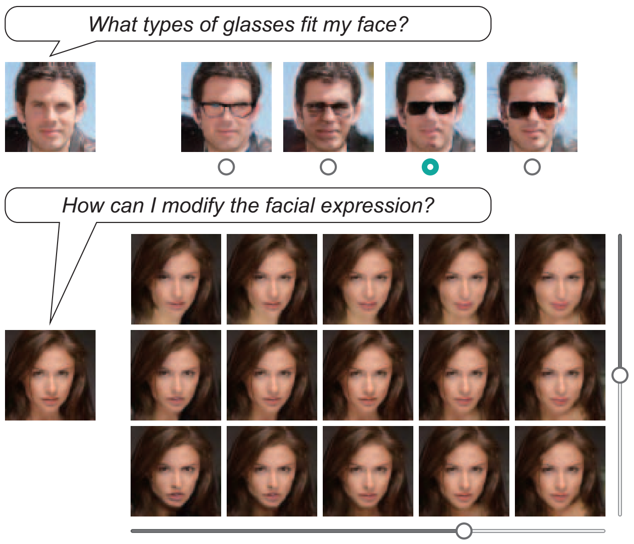
Takuhiro Kaneko Kaoru Hiramatsu Kunio Kashino
NTT Communication Science Laboratories, NTT Corporation
{kaneko.takuhiro, hiramatsu.kaoru, kashino.kunio}@lab.ntt.co.jp

Abstract

We present a generative attribute controller (GAC), a novel functionality for generating or editing an image while intuitively controlling large variations of an attribute. This controller is based on a novel generative model called the conditional filtered generative adversarial network (CFGAN), which is an extension of the conventional conditional GAN (CGAN) that incorporates a filtering architecture into the generator input. Unlike the conventional CGAN, which represents an attribute directly using an observable variable (e.g., the binary indicator of attribute presence) so its controllability is restricted to attribute labeling (e.g., restricted to an ON or OFF control), the CFGAN has a filtering architecture that associates an attribute with a multidimensional latent variable, enabling latent variations of the attribute to be represented. We also define the filtering architecture and training scheme considering controllability, enabling the variations of the attribute to be intuitively controlled using typical controllers (radio buttons and slide bars). We evaluated our CFGAN on MNIST, CUB, and CelebA datasets and show that it enables large variations of an attribute to be not only represented but also intuitively controlled while retaining identity. We also show that the learned latent space has enough expressive power to conduct attribute transfer and attribute-based image retrieval.

1. Introduction

In computer vision and machine learning, generative image modeling has been actively investigated to explore the "secret" behind images. In particular, an open issue has been to learn a latent space that has low dimensionality but is so expressive that we can extract realistic images from it. However, recent studies on generative image modeling with deep neural networks [12, 22, 40] have given us a clue to a solution. These studies presented promising results, indicating that their methods enable the learning of a latent space from which we can randomly generate images with high visual fidelity. However, it is not easy to extract a desired image from this space because the variables are highly entangled in the space, and the individual dimensions do not correspond to specific semantic features.


What types of glasses fit my face?
How can I modify the facial expression?
Figure 1. Practical examples of generative attribute controller. Our goal is to develop functionality for generating or editing image while intuitively controlling variations of attribute.

To solve this problem, we aim to learn a latent space that is not only expressive but also so controllable that a user can intuitively find and obtain a desired image. In particular, we focused on the controllability of attributes and developing a *generative attribute controller (GAC)* with which a user can generate or edit an image while intuitively controlling the variations of an attribute. Figure 1 shows practical examples of a GAC.

To develop a "good" GAC, we need to obtain a latent space that is (1) disentangled, (2) expressive, and (3) controllable. First, attributes and identity need to be disentangled in the latent space to change the attributes independently from the identity. For example, when modifying the facial expression in a portrait, a user wants to do so without compromising the person's identity. Second, the latent space needs to be expressive enough on attributes as to provide the attribute change that a user imagines. This is a challenging task because an "attribute" (e.g., "glasses") has many variations (e.g., sunglasses, round glasses, and thin glasses). Third, controllability is important because our goal is to enable a user to intuitively control attributes.

To satisfy these three requirements, we propose a generative model called the *conditional filtered generative adversarial network (CFGAN)*, which is an extension of the conditional GAN (CGAN) [11, 35] that incorporates a filtering architecture into the generator input. The CFGAN disentangles attributes and identity by learning an attribute-conditional generator and discriminator in an adversarial process. In fact, this learning scheme is the same as the CGAN, but we introduce a simple but powerful modification to the CGAN to obtain expressiveness and controllability. The CGAN represents each attribute using an observable variable (e.g., the binary indicator of attribute presence) so its controllability is restricted to attribute labeling (e.g., restricted to an ON or OFF control). In contrast, the CFGAN has a filtering architecture that associates each attribute with a multi-dimensional latent variable. This allows each attribute to be represented more expressively, i.e., multi-dimensionally. To achieve controllability, we developed three types of filtering architectures that enable a user to control attributes using typical controllers (radio buttons and slide bars). We evaluated our CFGAN on various types of data, i.e., digits (MNIST), birds (CUB), and faces (CelebA). These results show that our CFGAN can not only represent large variations of an attribute but change attributes while retaining identity. We also show that the learned latent space has enough expressive power to conduct attribute transfer and attribute-based image retrieval.

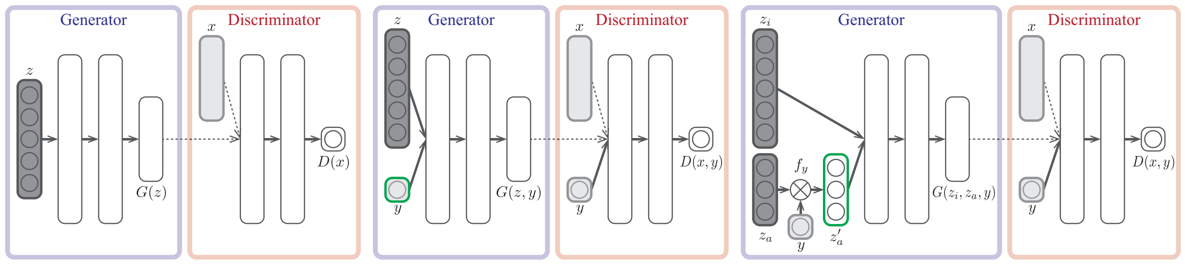
Contributions: Our contributions are summarized as follows. (1) We present a novel functionality called a GAC with which a user can generate and edit an image while intuitively controlling large variations of an attribute. (2) To learn a disentangled, expressive, and controllable latent space, we propose a deep generative model called the CFGAN. (3) The experimental evaluation indicates the controllability and disentanglement in attribute-based image generation and editing as well as the expressiveness in attribute transfer and attribute-based image retrieval.

2. Related Work

Image Editing: Image editing has been actively investigated in computer graphics, and various tasks have been tackled, e.g., from color modification (e.g., color transfer [39] and colorization [29]) to content modification (e.g., missing data interpolation [2] and image warping [1, 15]). There are two major approaches of attribute-based image editing: example-based [14, 30, 42, 48] and model-based [15, 20]. An example-based approach extracts an attribute-related patch from a reference image and transfers it to a target one. This enables attributes to be modified in various ways by using various reference images, but it requires images under special conditions (e.g., frontal faces [30], lightly made-up faces [14, 42], and reference images of the same persons [48]). A model-based approach constructs a model and modifies an image on the basis of the model. Model-based approaches handling unconstrained images have recently been proposed [15, 20], but they are task-specific and cannot be applied to arbitrary attributes. The reason these previous studies were limited is that they only obtained low-level information of images. In contrast, we use a deep generative model to obtain high-level information. Few studies [51, 4] have been conducted to attempt to edit an image using deep generative models. For these studies, user interaction in a low-level space (e.g., sketching in a photo) was assumed, but we assume this in a high-level space (e.g., controlling the value in the latent space).

Deep Generative Models: A large body of work exists on representation learning with deep generative models. Early studies were conducted to attempt to learn representation in a unsupervised fashion by using restricted Boltzmann machines or stacked auto-encoders [16, 17, 41, 44] and more recently stochastic neural networks [3, 13, 22, 40], adversarial networks [6, 12, 38], and autoregressive models [43]. In contrast, several studies were conducted to attempt to learn disentangled representation in a supervised fashion. Most studies used supervised data directly as input or output of the network [8, 11, 21, 34, 35, 47, 49, 52]. To the best of our knowledge, few studies [24, 33] have used supervised data to learn the latent variations of these data. Deep convolutional inverse graphics networks (DC-IGNs) [24] use a clamping technique to learn variations of graphics codes. They provide promising results, but their technique is difficult to apply to natural images because they use a graphics engine to obtain the training data. Adversarial autoencoders (AAEs) [33] incorporate supervised data in adversarial training to shape the distribution of the latent variables. The results showed that AAEs enable complicated distributions to be imposed, but applicable data are limited (e.g. gray scale or small images) because the training procedure was still not well established. In contrast, the CFGAN is a natural extension of a GAN and can be applied to complicated datasets based on recent progress in this area [5, 38].

Attribute Representation: In computer vision, how to represent attributes has been actively discussed. Early studies represented an attribute as a binary value indicating the presence or absence of the attribute [10, 25, 26]. However, for large variations of an attribute, this binary representation is not only restrictive but unnatural. To overcome this problem, a relative attribute indicating the strength of an attribute in an image with respect to others was developed



(a) GAN (b) CGAN (c) CFGAN

Figure 2. Differences in network architectures. Dark gray indicates latent variables, while light gray indicates observable variables. Variables surrounded with green lines can be used to control attribute. (a) In GAN, attribute is not explicitly represented, so its generator cannot be controlled on it. (b) In CGAN, attribute is represented using observable conditional variable y (e.g., binary indicator of attribute presence), so its controllability is restricted to attribute labeling (e.g., restricted to ON or OFF control). (c) In CFGAN, attribute is represented using multi-dimensional conditional latent variable z'_a, so its generator can be controlled more expressively, i.e., multi-dimensionally.

[37]. Considering the fact that some differences cannot be defined by the relative order, a representation indicating whether a given pair is distinguishable or not was recently proposed [50]. These previous studies suggested the complexity of attributes and the difficulty in defining a rule for organizing them. We expect our study to provide a clue to a solution since we enable interpretable latent variables representing large variations of an attribute to be learned without a detailed description of the attribute, i.e., only using the binary indicator of attribute presence.

3. Approach

In this section, we describe our CFGAN. We first describe the CGAN [11, 35], which is the basis of the CFGAN and then explain the formulation of the CFGAN, which integrates a filtering architecture into the CGAN.

3.1. Conditional Generative Adversarial Networks

The CGAN [11, 35] is an extension of a GAN [12] for conditional settings. We begin by briefly reviewing a GAN, followed by the formulation of the CGAN.

A GAN is a framework for training a generative model using a minmax game. It is composed of two networks: a *generator* G that maps a noise variable $z \sim P_z(z)$ to data space $x = G(z)$ and a *discriminator* D that assigns a probability $p = D(x) \in [0, 1]$ when x is a real training sample and assigns a probability $1 - p$ when x is generated by G. The $P_z(z)$ is a prior on z, and a uniform $[-1, 1]$ distribution is typically chosen. A minmax objective is used to train both networks together:

$$\begin{aligned}\min_G \max_D \; & \mathbb{E}_{x \sim P_{\text{data}}(x)}[\log D(x)] \\ & + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))]. \end{aligned} \tag{1}$$

This encourages D to find the binary classifier providing the best possible discrimination between real and generated data and simultaneously encourages G to fit the true data distribution. Both G and D can be trained with backpropagation. In this model, an attribute is not explicitly represented, so the generator output cannot be controlled on it as shown in Figure 2 (a).

The CGAN is an extension of a GAN, where G and D receive an additional variable y as input. The CGAN objective function can be rewritten as

$$\begin{aligned}\min_G \max_D \; & \mathbb{E}_{x, y \sim P_{\text{data}}(x, y)}[\log D(x, y)] \\ & + \mathbb{E}_{z \sim P_z(z), y \sim P_y(y)}[\log(1 - D(G(z, y), y))]. \end{aligned} \tag{2}$$

This model allows the generator output to be controlled by y, as shown in Figure 2 (b).

3.2. Conditional Filtered Generative Adversarial Networks

The conventional CGAN [11, 35] uses the additional variable y directly as input; therefore, the controllability of the generator is strongly restricted by the definition of y. For example, if y is a binary indicator of the presence of an attribute, the controllability is restricted to select ON or OFF. If we obtained detailed supervision of large variations of an attribute, we would be able to obtain the detailed controllability even using this model. However, it is difficult to do so since not only does it incur large annotation cost but the definition of attributes is not trivial [37, 50].

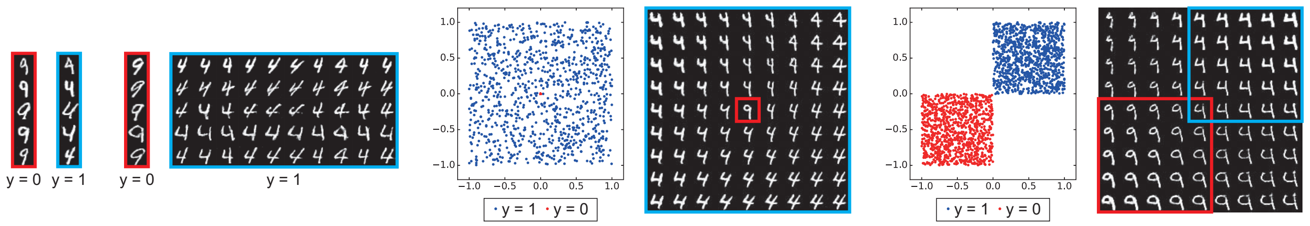
To solve this problem, our CFGAN integrates a filtering architecture into the CGAN. The CFGAN uses y to make a noise variable $z_a \sim P_{z_a}(z_a)$ conditioned on it.

$$z'_a = f_y(z_a), \tag{3}$$

where f_y is a filter function that maps z_a depending on y, and z'_a is a conditional latent variable that is fed to the generator input. The objective function thus becomes

$$\begin{aligned}\min_G \max_D \; & \mathbb{E}_{x, y \sim P_{\text{data}}(x, y)}[\log D(x, y)] \\ & + \mathbb{E}_{z_i \sim P_{z_i}(z_i), z_a \sim P_{z_a}(z_a), y \sim P_y(y)} \\ & \quad [\log(1 - D(G(z_i, z_a, y), y))], \end{aligned} \tag{4}$$

where z_i is an unconditional latent variable that has the same role as z in the CGAN. This model allows the generator output to be controlled by z'_a. For example, when



(a) CGAN (b) CFGAN + RB (c) CFGAN + SB-I (d) CFGAN + SB-II

Figure 3. Examples of attribute control using different models. We assume situation in which attribute "4" ($y = 1$, surrounded by blue lines) is added to non-attribute state "9" ($y = 0$, surrounded by red lines). In (a) and (b), row shows sample images generated from same z and z_i, respectively. In (b), column in $y = 1$ contains five samples from same category in z_a. These results indicate that CFGAN + RB allows user to select attribute while retaining style close to non-attribute state. In (c) and (d), left figures show sample data points in z'_a, and right figures show sample images generated from fixed z_i and varying z'_a. In (c), non-attribute state is centered and attribute is controlled on basis of it. In (d), attribute and non-attribute are controlled continuously.

we represent z'_a with a multi-dimensional variable, we can control the attribute multi-dimensionally. Figure 2 illustrates the differences in network architectures of the GAN, CGAN, and CFGAN. As shown in (b) and (c), the CFGAN uses the same observable variable as the CGAN, i.e., the CFGAN does not require additional supervision in training, but it can represent an attribute more expressively using z'_a.

4. Design & Training for Controller

4.1. Designing Controller

To use our CFGAN for a GAC, z'_a must be not only expressive but so controllable that a user can modify it intuitively. This controllability depends on the design of f_y and z_a. In this paper, we developed three architectures that allow a user to control attributes using typical controllers, i.e., radio buttons and slide bars.

Radio-Button Controller: A radio-button controller enables a user to select one from several choices. Each option is represented by one radio button, and a user can select only one in a radio-button group. In our case, a radio-button controller can be used to control the variations of an attribute categorically. For example, when a user wants to add glasses to a portrait photo, a radio-button controller allows him/her to select his/her favorite from several candidates such as sunglasses or transparent glasses. We model a radio-button controller (**RB**) using f_y that passes z_a when an attribute exists ($y = 1$) and z_a that follows categorical distribution.

$$f_y(z_a) = \begin{cases} z_a & (y = 1) \\ 0 & (y = 0) \end{cases}, z_a \sim \text{Cat}\left(K = k, p = \frac{1}{k}\right), \quad (5)$$

where k is the number of choices.

Slide-Bar Controller: A slide-bar controller enables a user to control values continuously. It consists of a scale and a "thumb" indicating the current value. In our case, a slide-bar controller can be used to control the variations of an attribute continuously. For example, when a user modifies the facial expression in a portrait photo, a slide-bar controller enables him/her to continuously modify it (e.g., from a slight smile to a broad smile). We model a slide-bar controller in two ways. We model one (**SB-I**) with f_y that passes z_a when an attribute exists ($y = 1$) and z_a that follows continuous uniform distribution.

$$f_y(z_a) = \begin{cases} z_a & (y = 1) \\ 0 & (y = 0) \end{cases}, z_a \sim \text{Unif}\,(-1, 1)\,. \quad (6)$$

The center of this slide bar indicates the non-attribute state and allows the variations of an attribute to be controlled on the basis of this state.

The other way (**SB-II**) is with f_y that projects z_a into the positive field when an attribute exists ($y = 1$) and projects z_a into the negative field when an attribute does not exist ($y = 0$).

$$f_y(z_a) = \begin{cases} |z_a| & (y = 1) \\ -|z_a| & (y = 0) \end{cases}, z_a \sim \text{Unif}\,(-1, 1)\,. \quad (7)$$

The negative field of this slide bar indicates the non-attribute state, and its positive field indicates the attribute state.

Figure 3 shows examples of attribute control with the CGAN, CFGAN + RB, CFGAN + SB-I, and CFGAN + SB-II. We used the two types of digits "4" (attribute) and "9" (non-attribute) in the MNIST dataset. The experimental setup is described in detail in Sec. 6.

Relationship between CGAN and CFGAN: We can see the CGAN as a special case of the CFGAN. When we use the CFGAN + RB where the number of choices $k = 1$ (i.e., the variations of an attribute are not selectable), it has the same architecture as the CGAN. Thus, the CFGAN can be seen as a natural extension of the CGAN.

4.2. Training for Controller

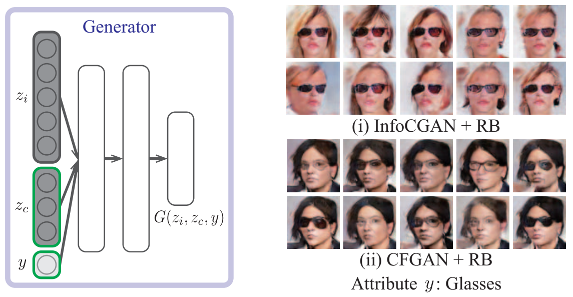
Analogous with the CGAN, the CFGAN can disentangle latent variables into a conditional one, z'_a, and unconditional one, z_i, by training the generator with the adversarial conditional discriminator. However, the CFGAN formulation in Eqn. 4 imposes no restrictions on the relationship of the elements of z'_a. Thus, it is possible that z'_a will be used by the generator in a highly entangled manner. This property is undesirable for using z'_a for a controller.

To avoid this, we extend an information-theoretic regularization [5] to the conditional setting and optimize the CFGAN objective with it. This regularization imposes a restriction on each element of z'_a to represent a different semantic feature of an attribute. In particular, we maximize the conditional mutual information between z'_a and $G(z_i, z_a, y)$: $I(z'_a; G(z_i, z_a, y)|y)$. In practice, this information is difficult to maximize directly because doing so requires calculation of the intractable posterior $P(z'_a|x, y)$. Following the study of Chen et al. [5], we instead calculate its lower bound with an auxiliary distribution $Q(z'_a|x, y)$ approximating $P(z'_a|x, y)$:

$$\begin{aligned}
&I(z'_a; G(z_i, z_a, y)|y) \\
=&H(z'_a|y) - H(z'_a|G(z_i, z_a, y), y) \\
=&H(z'_a|y) + \mathbb{E}_{x\sim G(z_i,z_a,y)}[\mathbb{E}_{\hat{z}'_a\sim P(z'_a|x,y)}[\log P(\hat{z}'_a|x, y)]] \\
=&H(z'_a|y) + \mathbb{E}_{x\sim G(z_i,z_a,y)}[D_{\mathrm{KL}}(P(\cdot|x, y)||Q(\cdot|x, y) \\
&+ \mathbb{E}_{\hat{z}'_a\sim P(z'_a|x,y)}[\log Q(\hat{z}'_a|x, y)]] \\
\geqq&H(z'_a|y) + \mathbb{E}_{x\sim G(z_i,z_a,y)}[\mathbb{E}_{\hat{z}'_a\sim P(z'_a|x,y)}[\log Q(\hat{z}'_a|x, y)]] \\
=&H(z'_a|y) + \mathbb{E}_{z'_a\sim P(z'_a|y), x\sim G(z_i,z'_a)}[\log Q(z'_a|x, y)]. \quad (8)
\end{aligned}$$

In the last term, we rewrite $G(z_i, z_a, y)$ as $G(z_i, z'_a)$ for ease of understanding. In this paper, for simplicity, we fix the latent variable distribution, i.e., treat $H(z'_a|y)$ as a constant. In training the CFGAN, we maximize the last term of Eqn. 8 with Eqn. 4.

Difference between CFGAN and InfoCGAN: In fact, an information-theoretic regularization similar to the above can be integrated into the CGAN (we call this model InfoCGAN). Figure 4 (a) shows the network architecture of this model. Its generator input is composed of an incontrollable variable z_i, controllable variable z_c, and conditional variable y, and an information-theoretic regularization is imposed on z_c. This enables the generator output to be controlled multi-dimensionally using z_c, like with the CFGAN, but the difference between the InfoCGAN and CFGAN is whether the controllable variable (z'_a in the CFGAN and z_c in the InfoCGAN) is conditioned on y. In the CFGAN, z'_a is conditioned on y using the filtering architecture, enabling the attribute-dependent features (e.g., sub-categories of glasses in the case of Figure 4 (b-ii)) to be obtained. In the InfoCGAN, however, the controllable variable z_c is independent from y, so the attribute-independent features (e.g., hair style and poses in the case of Figure 4 (b-i)) are obtained. The controllability of the InfoCGAN on attributes depends on not z_c but y and is similar to that of the CGAN.



(a) InfoCGAN (b) Example results

Figure 4. Network architecture of InfoCGAN (see Figure 2 for comparison) and example results from using InfoCGAN and CFGANs. In (b-i), sample images are generated from same z_i, same $y = 1$, and varying z_c using InfoCGAN with ten-categorical RB. In (b-ii), sample images are generated from same z_i and varying z'_a using CFGAN with ten-categorical RB.

5. Use of Controller

5.1. Latent Variable Estimation

In the CFGAN, attribute control is conducted in the latent space. Therefore, to edit the attributes of a given photo x using the CFGAN, we need to estimate the latent variables z_i and z'_a from x. To solve this problem, we first estimate y from x, next estimate z'_a from x and y, and finally estimate z_i from z'_a and x.

If obtaining x and y, we can estimate z'_a using the approximate posterior $Q(z'_a|x, y)$ in Eqn. 8; therefore, we first estimate y from x as follows:

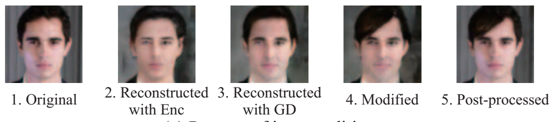
$$y^* = \arg\max_y P(y|x). \quad (9)$$

This stands for a standard classification task. Thus, we train the classifier $C(x)$ to predict the conditional distribution $P(y|x)$ using the pair training data x and y that are also used to train the CFGAN. We then estimate z'_a from x and y using $Q(z'_a|x, y)$.

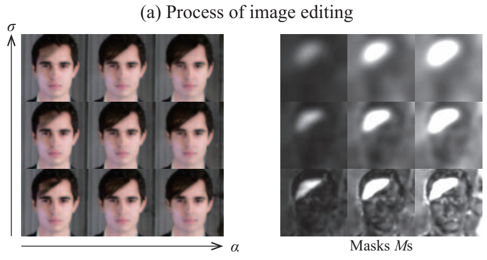
We next estimate z_i from z'_a and x using the manifold projection method [51]:

$$z_i^* = \arg\min_{z_i} \mathcal{L}(G(z_i, z'_a), x), \quad (10)$$

where $\mathcal{L}(x_1, x_2)$ is the distance metric between two images and we rewrite $G(z_i, z_a, y)$ as $G(z_i, z'_a)$ for ease of understanding. Following the recent success in perceptual losses [7, 19], we define $\mathcal{L}(x_1, x_2)$ using the distance in the features consisting of raw pixels and those extracted from the discriminator. Following previous research [51], we optimize Eqn. 10 in a two-stage manner. In fact, this function consists of differentiable neural networks, so we can directly optimize it using the gradient method. However, it is a non-convex problem; therefore, the reconstruction quality depends on the initialization of z_i. To obtain a "good" initialization, we train an encoder $E(x)$ that directly predicts z_i from x. We optimize it by minimizing the objective



(a) Process of image editing



(b) Post-processing with different masks Ms

Figure 5. Examples of image editing. (a) Given original image (step 1), it is first reconstructed with encoder (Enc) (step 2). It is used as initialization for gradient descent (GD)-based optimization (step 3). Its attribute (bangs) is modified with CFGAN (step 4). Post-processing is applied to it to remove small reconstruction error (step 5). (b) Post-processing with masks Ms where α and σ are different.

function $\mathcal{L}(G(E(x), z'_a), x)$. Given a novel image, we first estimate z_i using $E(x)$ then use it as the initialization for the gradient descent-based optimization. This not only reduces the reconstruction error but makes it fast enough to reach the optimum. Figure 5 (a) shows example images: original, reconstructed with the encoder, and reconstructed with the gradient descent. Two-step optimization slightly improves visual quality (e.g., eyebrows are recovered).

5.2. Post-processing

In the above reconstruction process, the image information is once abstracted into the low-dimensional space. This makes it difficult to reconstruct such an image that has pixel-wise perfect accuracy. This property is undesirable for using our CFGAN as an image editor.

To alleviate this effect, we conduct post-processing. We extend a masking technique [4] to apply to our CFGAN:

$$\tilde{x} = x_{\text{rec}} + M\Delta + (1 - M)\Delta'$$
$$\Delta = x_{\text{mod}} - x_{\text{rec}}, \Delta' = x - x_{\text{rec}}, \quad (11)$$

where x is the original image, x_{rec} is the reconstructed image, and x_{mod} is the modified and reconstructed image. The mask M switches between Δ and Δ' depending on the channel-wise mean of the absolute value of Δ. In previous study [4], it was smoothed with a standard Gaussian filter and truncated between 0 and 1. From intense trials, we found in our case that smoothing with a scaled Gaussian filter $\alpha \cdot g(\cdot)$ works well:

$$M = \min(\alpha \cdot g(|\overline{\Delta}|; \sigma), 1), \quad (12)$$

where α is a scale parameter, indicating the degree of how small change is captured, and σ is the standard deviation for a Gaussian kernel, indicating the degree of how smoothly attribute and non-attribute states are mixed. Since this post-processing is simple, a user can control the parameters interactively. Figure 5 shows images post-processed with masks Ms where α and σ are different. Note that this post-processing technique is simple but not restricted to the CFGAN and can be used for general generative models.

6. Experiments

We first describe the experimental setup then four main applications: attribute-based image generation, attribute-based image editing, attribute transfer, and attribute-based image retrieval. Please refer to the supplementary material for more results and comparisons.

Datasets: We evaluated our CFGAN on various image datasets: digits (MNIST [28]), birds (CUB [45]), and faces (CelebA [31]). The MNIST dataset consists of photos of handwritten digits, containing 60,000 training and 10,000 test samples. We used this dataset to clarify the basic characteristics of our CFGAN. The CUB dataset consists of photos of 200 bird species, containing 6,000 training and 6,000 test samples. We used the cropped version [47] and rescaled the images to 64×64. The CelebA dataset is an image dataset with photos of faces, containing 180,000 training and 20,000 test samples. We used the aligned and cropped version and scaled and cropped the images to 64×64. To avoid overfitting, we augmented the training data in CUB and CelebA with common operations [9, 23].

Implementation Details: To stabilize CFGAN training, we designed the two network architectures (D and G) on the basis of current techniques in the DCGAN [38] and InfoGAN [5]. The CFGAN mainly builds on multiple convolution and deconvolution layers, uses ReLU activation [36] and leaky ReLU activation [32, 46] in the generator and discriminator, respectively, and batch normalization [18] in both networks. Following the InfoGAN [5], we parameterized the auxiliary distribution Q in Eqn. 8 as a neural network. In particular, we made D and Q share all convolutional layers to reduce calculation cost. For a radio-button controller (RB), we represent $Q(z'_a|x, y)$ as softmax nonlinearity. For a slide-bar controller (SB-I and SB-II), we represent $Q(z'_a|x, y)$ as a factored Gaussian. We modeled the classifier C and encoder E as the same architecture as D except for the last layer. In particular, we made C and E share all convolutional layers. We describe the detailed experimental setup in the supplementary material.

Baseline: To clarify the characteristics of the CFGAN, we compared it with a GAN, which has no controllability on attributes, and the CGAN, which has only one-dimensional controllability. There are other deep generative models such as CVAE [47] and VAE/GAN + visual attribute vectors [27], but they represent an attribute as a one-dimensional vector, so their controllability is similar to that of the CGAN.

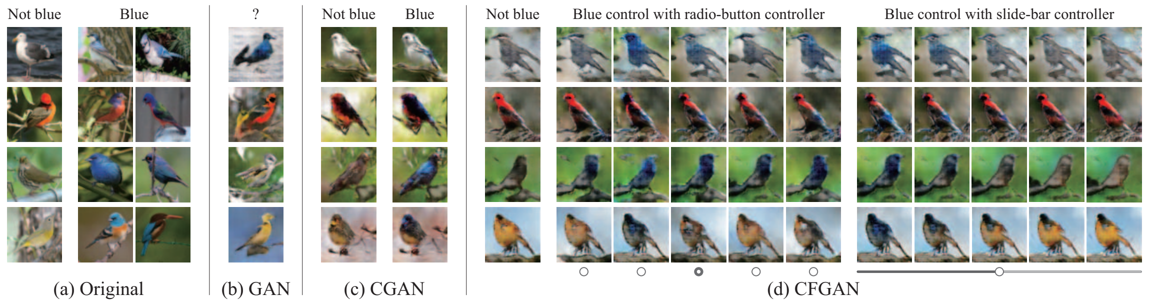

Not blue
Blue
?
Not blue
Blue
Not blue
Blue control with radio-button controller
Blue control with slide-bar controller
(a) Original
(b) GAN
(c) CGAN
(d) CFGAN

Figure 6. Example results of blue-bird generator. In (c) and (d), row contains samples generated from same attribute-independent variables z and z_i, respectively; column shows generated images for same attribute-dependent variables y and z'_a, respectively.

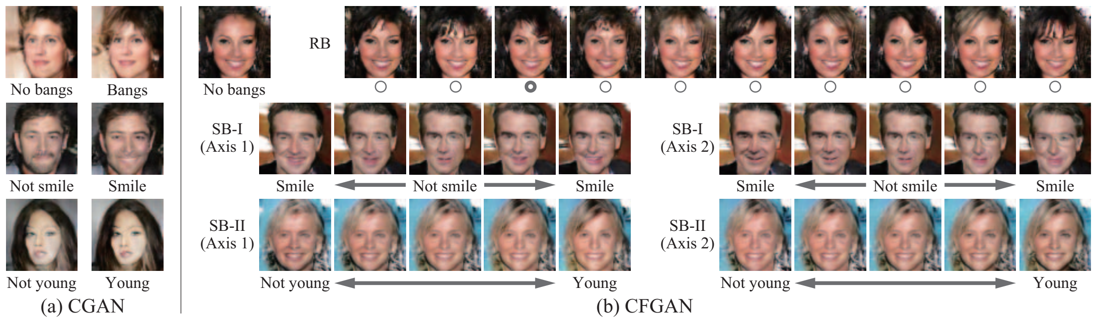


Figure 7. Example results of face-attribute generator. In (a), row contains samples generated from same z while varying y. In (b), row contains samples generated from same z_i while varying z'_a.

As discussed in Sec. 4.2, InfoCGAN also represents an attribute as a one-dimensional vector, but to further clarify the difference between the CFGAN and InfoCGAN, we provide an extended analysis in the supplementary material.

6.1. Attribute-based Image Generation

Control between Two Digits: To clarify the basic characteristics of the CFGAN, we used the subsets of the MNIST dataset. We selected two types of digits and regarded one as an attribute ($y = 1$) and the other as a non-attribute ($y = 0$). Example results are shown in Figure 3, where we regarded "4" as the attribute and "9" as the non-attribute. These results indicate that the CFGAN not only represents large variations of an attribute but enables a user to control the variations in various ways by using the different controllers.

Blue-Bird Generator: As shown in Figure 6 (a), "blue bird" covers many variations, and it is challenging to model them. We evaluated whether the CFGAN can learn variations of an attribute even in such a difficult situation. We selected the attributes with "blue" in the name (e.g., blue wing, blue eye, and blue leg) from 312 attributes, summarized all as blue bird, and used them as the supervised data, i.e., regard "blue" as the attribute ($y = 1$) and "not blue" as the non-attribute ($y = 0$). Figure 6 shows example results. We used the CFGAN with the combination of a five-categorical RB and a one-dimensional SB-I. The CGAN could give color to a non-attribute image (i.e., not-blue bird) by switching y but could not control how to colorize it. In contrast, the CFGAN could give color in various ways by varying z'_a categorically and continuously.

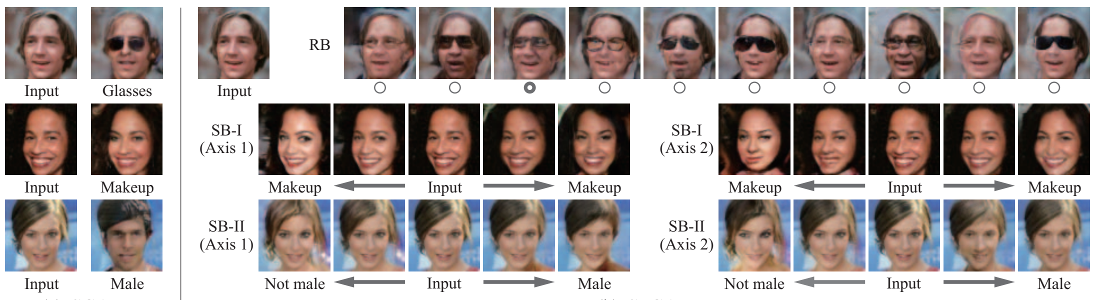
Face-Attribute Generator: To clarify the difference in controllers for various attributes, we implemented three controllers (a ten-categorical RB, three-dimensional SB-I, and three-dimensional SB-II) for six attributes (bangs, eyeglasses, heavy makeup, male, smiling, and young). Due to space limitations, we discuss some cases here and provide others in the supplementary material. Figure 7 shows example results. In the CGAN, an attribute was modified in one way, while the CFGAN enabled it to be controlled in various ways.

6.2. Attribute-based Image Editing

To evaluate the effectiveness of the CFGAN for editing an existing image, we applied the same face-attribute generators described in Sec. 6.1 to the image-editing task. Figure 8 shows example results. We discuss the images where the post-processing was applied. In both the CGAN and CFGAN, the attributes could be changed while retaining identities, but the controllability was different.

6.3. Attribute Transfer

By taking the advantage of the disentangled and expressive latent space of the CFGAN, we applied it to the attribute transfer and attribute-based image retrieval tasks. We describe these tasks in this and the next section, respectively. To conduct attribute-based image transfer, we first extracted z_i and z'_a from the target and the reference images, respectively. We then generated a modified image



(a) CGAN (b) CFGAN

Figure 8. Example results of attribute-based image editing. View of figure is similar to that in Figure 7, except that latent variables are extracted from given images.

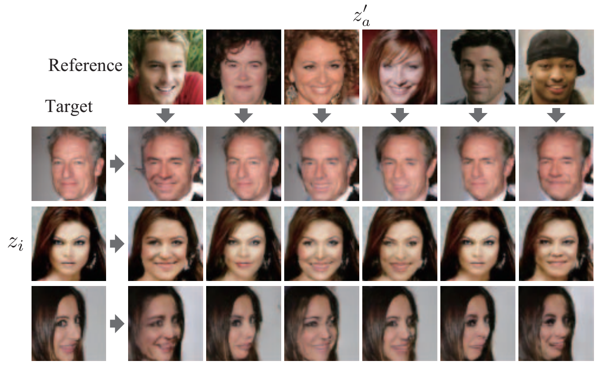
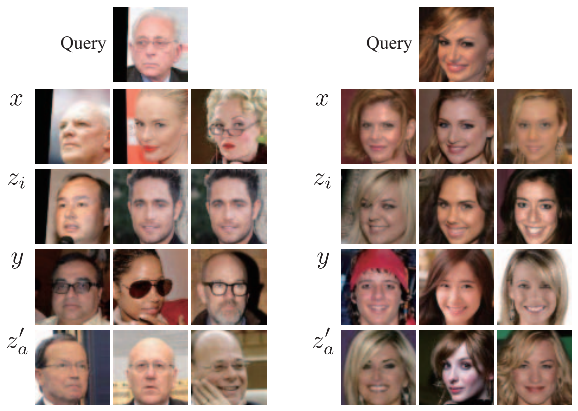


Figure 9. Example results of attribute transfer. Images are generated from z_i extracted from first-column images and z'_a extracted from first-row images.



(a) Eyeglasses-based retrieval (b) Bangs-based retrieval

Figure 10. Example results of attribute-based image retrieval. First row shows query images. Other rows show top three retrievals. From top to bottom, we measured distance in x, z_i, y, and z'_a.

from them. Figure 9 shows example results. We used the CFGAN with a three-dimensional SB-I for this task. In these results, the types of smiles (e.g., how to open the mouth) were transferred from the reference images to the target ones regardless of gender, age, and pose. Note that previous studies [27, 38] required multiple samples to do attribute transfer because their latent variables are highly entangled and they need to take the average to know the "secret" of the attribute, but we can do "one-shot" transfer by exchanging only z'_a because our latent variables are disentangled into attribute-dependent and attribute-independent.

6.4. Attribute-based Image Retrieval

For image retrieval, we searched for the image close to the query image in terms of Euclidian distance. For comparison, we measured the distance in the low-level image space x, attribute-label space y, attribute-independent latent variable space z_i, and attribute-dependent latent variable space z'_a. We predicted y, z_i, and z'_a using C, E, and Q, respectively. Figure 10 shows example results. We used the CFGAN with a three-dimensional SB-I for this task. When an image was searched based on x or z_i, the images that were globally similar to the query, including the background, were retrieved. When an image was searched based on y or z'_a, the images that had the same attribute as the query were retrieved. In particular, when an image was searched based on z'_a, the images that had the "same" type of attribute (e.g., thin glasses in (a) and left parted hair in (b)) were retrieved. Note that we used only the supervised data indicating the presence or absence of the attribute and learned the type of attribute in an unsupervised fashion.

7. Discussion and Conclusion

We introduced a GAC, a novel functionality for generating or editing an image while intuitively controlling large variations of an attribute. To develop it, we proposed the CFGAN and we defined the design and training scheme to use it for a controller. We obtained promising results, indicating that the learned latent space is disentangled, controllable, and expressive. One limitation is that it is impossible to name each dimension of z'_a in advance since it is learned in an unsupervised fashion. However, the visual-attribute change is consistent for the same z'_a, so it can be named after learning. Moreover, when there are so many latent variations that experts cannot name them all, the CFGAN is expected to propose a new explanation without detailed supervision. Possible extensions to this study include developing a filtering architecture representing other types of controllers, applying our filtering technique to other generative models, and using the CFGAN as a latent variation-discovery tool for high-dimensional data.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. In *SIGGRAPH*, 2000. 2

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[3] Y. Bengio, É. Thibodeau-Laufer, G. Alain, and J. Yosinski. Deep generative stochastic networks trainable by backprop. In *ICML*, 2014. 2

[4] A. Brock, T. Lim, J. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 2, 6

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016. 2, 5, 6

[6] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[7] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *NIPS*, 2016. 5

[8] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 2

[9] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 6

[10] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2

[11] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014. 2, 3

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 1, 2, 3

[13] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015. 2

[14] D. Guo and T. Sim. Digital face makeup by example. In *CVPR*, 2009. 2

[15] T. Hassner, S. Harel, E. Paz, and R. Enbar. Effective face frontalization in unconstrained images. In *CVPR*, 2015. 2

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006. 2

[17] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 2

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 6

[19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 5

[20] I. Kemelmacher-Shlizerman, S. Suwajanakorn, and S. M. Seitz. Illumination-aware age progression. In *CVPR*, 2014. 2

[21] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014. 2

[22] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR*, 2014. 1, 2

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 6

[24] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015. 2

[25] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, 2009. 2

[26] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 2

[27] A. B. L. Larsen, S. K. Sønderby, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 6, 8

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998. 6

[29] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *ACM Trans. Graph.*, volume 23, pages 689–694, 2004. 2

[30] L. Liu, H. Xu, J. Xing, S. Liu, X. Zhou, and S. Yan. Wow! You are so beautiful today! In *ACMMM*, 2013. 2

[31] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015. 6

[32] A. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 6

[33] A. Makhzani, J. Shlens, N. Jaitly, and I. Goodfellow. Adversarial autoencoders. In *NIPS*, 2016. 2

[34] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov. Generating images from captions with attention. In *ICLR*, 2016. 2

[35] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2, 3

[36] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010. 6

[37] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, 2011. 3

[38] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 2, 6, 8

[39] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Comput. Graph.*, 21:34–41, 2001. 2

[40] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. 1, 2

[41] R. Salakhutdinov and G. E. Hinton. Deep Boltzmann machines. In *AISTATS*, 2009. 2

[42] W.-S. Tong, C.-K. Tang, M. S. Brown, and Y.-Q. Xu. Example-based cosmetic transfer. In *PG*, 2007. 2

[43] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. 2

[44] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, 2008. 2

[45] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011. 6

[46] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. In *ICML Workshop*, 2015. 6

[47] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2Image: Conditional image generation from visual attributes. In *ECCV*, 2016. 2, 6

[48] F. Yang, J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas. Expression flow for 3D-aware face component transfer. In *SIGGRAPH*, 2011. 2

[49] J. Yim, H. Jung, B. Yoo, C. Choi, D. Park, and J. Kim. Rotating your face using multi-task deep neural network. In *CVPR*, 2015. 2

[50] A. Yu and K. Grauman. Just noticeable differences in visual attributes. In *ICCV*, 2015. 3

[51] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2, 5

[52] Z. Zhu, P. Luo, X. Wang, and X. Tang. Multi-view perceptron: A deep model for learning face identity and view representations. In *NIPS*, 2014. 2