Disentangling Latent Space for VAE by Label Relevant/Irrelevant Dimensions

Zhilin Zheng[1] Li Sun[1]
[1] Shanghai Key Laboratory of Multidimensional Information Processing,
East China Normal University
51171214020@stu.ecnu.edu.cn sunli@ee.ecnu.edu.cn

Abstract

VAE requires the standard Gaussian distribution as a prior in the latent space. Since all codes tend to follow the same prior, it often suffers the so-called "posterior collapse". To avoid this, this paper introduces the class specific distribution for the latent code. But different from cVAE, we present a method for disentangling the latent space into the label relevant and irrelevant dimensions, $\mathbf{z}_s$ and $\mathbf{z}_u$, for a single input. We apply two separated encoders to map the input into $\mathbf{z}_s$ and $\mathbf{z}_u$ respectively, and then give the concatenated code to the decoder to reconstruct the input. The label irrelevant code $\mathbf{z}_u$ represent the common characteristics of all inputs, hence they are constrained by the standard Gaussian, and their encoder is trained in amortized variational inference way, like VAE. While $\mathbf{z}_s$ is assumed to follow the Gaussian mixture distribution in which each component corresponds to a particular class. The parameters for the Gaussian components in $\mathbf{z}_s$ encoder are optimized by the label supervision in a global stochastic way. In theory, we show that our method is actually equivalent to adding a KL divergence term on the joint distribution of $\mathbf{z}_s$ and the class label c, and it can directly increase the mutual information between $\mathbf{z}_s$ and the label c. Our model can also be extended to GAN by adding a discriminator in the pixel domain so that it produces high quality and diverse images.

1. Introduction

Learning a deep generative model for the structured image data is difficult because this task is not simply modeling a many-to-one mapping function such as the classification, instead it is often required to generate diverse outputs for similar codes sampled from a simple distribution. Furthermore, image $\mathbf{x}$ in the high dimension space often lies in a complex manifold, thus the generative model should capture the underlying data distribution $p(\mathbf{x})$.

Basically, Variational Auto-Encoder (VAE) [34, 20] and Generative Adversarial Network (GAN) [13, 25] are two strategies for structured data generation. In VAE, the encoder $q_\phi(\mathbf{z}|\mathbf{x})$ maps data $\mathbf{x}$ into the code $\mathbf{z}$ in latent space. The decoder, represented by $p_\theta(\mathbf{x}|\mathbf{z})$, is given a latent code $\mathbf{z}$ sampled from a distribution specified by the encoder and tries to reconstruct $\mathbf{x}$. The encoder and decoder in VAE are trained together mainly based on the data reconstruction loss. At the same time, it requires to regularize the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to be simple (*e.g.* Gaussian) based on the Kullback-Leibler (KL) divergence between $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$, so that the sampling in latent space is easy. Optimization for VAE is quite stable, but results from it are blurry. Mainly because the posterior defined by $q_\phi(\mathbf{z}|\mathbf{x})$ is not complex enough to capture the true posterior, also known for "posterior collapse". On the other hand, GAN treats the data generation task as a min/max game between a generator $G(\mathbf{z})$ and discriminator $D(\mathbf{x})$. The adversarial loss computed from the discriminator makes generated image more realistic, but its training becomes more unstable. In [10, 22, 28], VAE and GAN are integrated together so that they can benefit each other.

Both VAE and GAN work in an unsupervised way without giving any condition of the label on the generated image. Instead, conditional VAE (cVAE) [39, 3] extends it by showing the label c for both encoder and decoder. It learns data distribution conditioned on the given label. Hence, the encoder and decoder become $q_\phi(\mathbf{z}|\mathbf{x}, c)$ and $p_\theta(\mathbf{x}|\mathbf{z}, c)$. Similarly, in conditional GAN (cGAN) [9, 18, 33, 30] label c is given to both generator $G(\mathbf{z}, c)$ and discriminator $D(\mathbf{x}, c)$. Theoretically, feeding label c to either the encoder in VAE or decoder in VAE or GAN helps increasing the mutual information between the generated $\mathbf{x}$ and the label c. Thus, it can improve the quality of generated image.

This paper deals with image generation problem in VAE with two separate encoders. For a single input $\mathbf{x}$, our goal is to disentangle the latent space code $\mathbf{z}$, computed by encoders, into the label relevant dimensions $\mathbf{z}_s$ and irrelevant ones $\mathbf{z}_u$. We emphasize the difference between $\mathbf{z}_s$ and $\mathbf{z}_u$, and their corresponding encoders. For $\mathbf{z}_s$, since label c is known during training, it should be more accurate and specific. While without any label constraint, $\mathbf{z}_u$ should be gen-

eral. Specifically, the two encoders are constrained with different priors on their posterior distributions $q_{\phi_s}(\mathbf{z}_s|\mathbf{x})$ and $q_{\phi_u}(\mathbf{z}_u|\mathbf{x})$. Similar with VAE or cVAE, in which the full code $\mathbf{z}$ is label irrelevant, the prior for $\mathbf{z}_u$ is also chosen $\mathcal{N}(0, \mathbf{I})$. But different from previous works, the prior $p(\mathbf{z}_s)$ becomes complex to capture the label relevant distribution. From the decoder's perspective, it takes the concatenation of $\mathbf{z}_s$ and $\mathbf{z}_u$ to reconstruct the input $\mathbf{x}$. Here the distinction with cVAE and cGAN is that they uses the fixed, one-hot encoding label, while our work applies $\mathbf{z}_s$, which is considered to be a variational, soft label.

Note that there are two stages for training our model. First, the encoder for $\mathbf{z}_s$ gets trained for classification task under the supervision of label c. Here instead of the softmax cross entropy loss, Gaussian mixture cross entropy loss proposed in [44] is adopted since it accumulates the mean $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\sigma}_c$ for samples with the same label c, and models it as the Gaussian $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$, hence $\mathbf{z}_s \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$. The first stage specifies the label relevant distribution. In the second stage, the two encoders and the decoder are trained jointly in an end-to-end manner based on the reconstruction loss. Meanwhile, priors of $\mathbf{z}_s \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$ and $\mathbf{z}_u \sim \mathcal{N}(0, \mathbf{I})$ are also considered.

The main contribution of this paper lies in following aspects: (1) for a single input $\mathbf{x}$ to the encoder, we provide an algorithm to disentangle the latent space into label relevant and irrelevant dimensions in VAE. Previous works like [15, 4, 37] disentangle the latent space in AE not VAE. So it is impossible to make the inference from their model. Moreover, [27, 4, 23] requires at least two inputs for training. (2) we find the Gaussian mixture loss function is suitable way for estimating the parameters of the prior distribution, and it can be optimized in VAE framework. (3) we give both a theoretical derivation and a variety of detailed experiments to explain the effectiveness of our work.

2. Related works

Two types of methods for the structured image generation are VAE and GAN. VAE [20] is a type of parametric model defined by $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$, which employs the idea of variational inference to maximize the evidence lower bound (ELBO), as is shown in (1).

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{1}$$

The right side of the above is the ELBO, which is the lower bound of maximum likelihood. In VAE, a differentiable encoder-decoder are connected, and they are parameterized by ϕ and θ, respectively. $E_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z}))$ represents the end-to-end reconstruction loss, and $\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$ is the KL divergence between the encoder's output distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p(\mathbf{z})$, which is usually modeled by standard normal distribution $\mathcal{N}(0, \mathbf{I})$. Note that VAE assumes that the posterior $q_\phi(\mathbf{z}|\mathbf{x})$ is of Gaussian, and the $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are estimated for every single input $\mathbf{x}$ by the encoder. This strategy is named amortized variational inference (AVI), and it is more efficiency than stochastic variational inference (SVI) [17].

VAE's advantage is that its loss is easy to optimize, but the simple prior in latent space may not capture the complex data patterns which often leads to the mode collapse in latent space. Moreover, VAE's code is hard to be interpreted. Thus, many works focus on improving VAE on these two aspects. cVAE [39] adds the label vector as the input for both the encoder and decoder, so that the latent code and generated image are conditioned on the label, and potentially prevent the latent collapse. On the other hand, β-VAE [16, 7] is a unsupervised approach for the latent space disentanglement. It introduces a simple hyper-parameter β to balance the two loss term in (1). A scheme named infinite mixture of VAEs is proposed and applied in semi-supervised generation [1]. It uses multiple number of VAEs and combines them as a non-parametric mixture model. In [19], the semi-amortized VAE is proposed. It combines AVI with SVI in VAE. Here the SVI estimates the distribution parameters on the whole training set, while the AVI in traditional VAE gives this estimation for a single input.

GAN [13] is another technique to model the data distribution $p_D(\mathbf{x})$. It starts from a random $\mathbf{z} \sim p(\mathbf{z})$, where $p(\mathbf{z})$ is simple, *e.g.* Gaussian, and trains a transform network $g_\theta(\mathbf{z})$ under the help of discriminator $D_\phi(\cdot)$ so that $p_\theta(\mathbf{z})$ approximates $p_D(\mathbf{x})$. The later works [32, 26, 2, 14, 29] try to stabilize GAN's training. Traditional GAN works in a fully supervised manner, while cGAN [18, 33, 30, 6] aims to generate images conditioned on labels. In cGAN, the label is given as an input to both the generator and discriminator as a condition for the distribution. The encoder-decoder architecture like AE or VAE can also be used in GAN. In ALI [11] and BiGAN [10], the encoder maps $\mathbf{x}$ to $\mathbf{z}$, while the decoder reverses it. The discriminator takes the pair of $\mathbf{z}$ and $\mathbf{x}$, and is trained to determine whether it comes from the encoder or decoder in an adversarial manner. In VAE-GAN [22, 24], VAE's generated data are improved by a discriminator. Similar idea also applies to cVAE in [3]. VAE-GAN also applies in some specific applications like [4, 12].

Since code $\mathbf{z}$ potentially affects the generated data, some works try to model its effect and disentangle the dimensions of $\mathbf{z}$. InfoGAN [9] reveals the effect of latent space code c by maximizing the mutual information between c and the synthetic data $g_\theta(\mathbf{z}, \mathbf{c})$. Its generator outputs $g_\theta(\mathbf{z}, c)$ which is inspected by the discriminator $D_\phi(\cdot)$. $D_\phi(\cdot)$ also tries to reconstruct the code c. In [27], the latent dimension is disentangled in VAE based on the specified factors and unspecified ones, which is similar with our work. But its encoder takes multiple inputs, and the decoder combines codes from different inputs for reconstruction. The work in [15] mod-

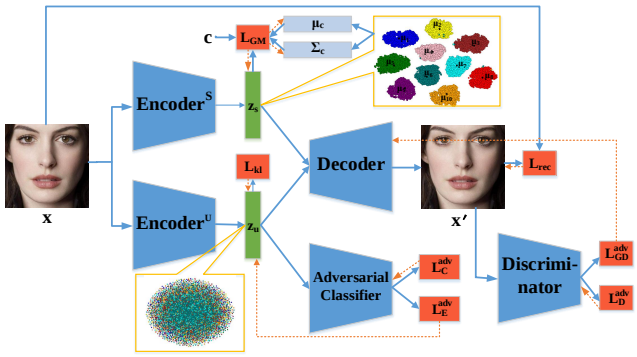


Figure 1. **The network architecture**. We disentangle class relevant dimensions $\mathbf{z}_s$ and class irrelevant dimensions $\mathbf{z}_u$ in the latent space. The $Encoder^s$ maps input image $\mathbf{x}$ to $\mathbf{z}_s$, and forces $\mathbf{z}_s$ to be well classified while following a Gaussian mixture distribution with learned mean $\boldsymbol{\mu}_c$ and covariance $\boldsymbol{\Sigma}_c$. Meanwhile, the $Encoder^u$ extracts $\mathbf{z}_u$ from $\mathbf{x}$ and pushes it to match the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The adversarial classifier is added on the top of $\mathbf{z}_u$ to distinguish the class of $\mathbf{z}_u$, while $Encoder^u$ tries to fool it. Then $\mathbf{z}_s$ and $\mathbf{z}_u$ are concatenated and fed into the $Decoder$ to obtain $\mathbf{x}'$ for reconstruction. The adversarial training in the pixel domain is also adopted with a discriminator added on the images. The forward pass process is drawn in solid lines and dashed lines represent back propagation.

ifies [27] by taking a single input. To stabilize training, its model is built in AE not VAE, hence it can't perform variational inference. Other works in [37, 4, 23] are also built in AE and more than two inputs. Moreover they only apply in a particular domain like face [37, 4] or image-to-image translation [23], while our work is built in VAE and takes only a single input for a more general case.

3. Proposed method

We propose a image generation algorithm based on VAE which divides the encoder into two separate ones, one encoding label relevant representation $\mathbf{z}_s$ and the other encoding label irrelevant information $\mathbf{z}_u$. $\mathbf{z}_s$ is learned with supervision of the categorical class label and it is required to follow a Gaussian mixture distribution, while $\mathbf{z}_u$ is wished to contain other common information irrelevant to the label and is made close to standard Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

3.1. Problem formulation

Given a labeled dataset $\mathcal{D}_s = \{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$, where $\mathbf{x}^{(i)}$ is the i-th images and $y^{(i)} \in \{0, 1, \cdots, C-1\}$ is the corresponding label. C and N are the number of classes and the size of the dataset, respectively. The goal of VAE is to maximum the ELBO defined in (1), so that the data log-likelihood $\log p(\mathbf{x})$ is also maximized. The key idea is to split the full latent code $\mathbf{z}$ into the label relevant dimensions $\mathbf{z}_s$ and the irrelevant dimensions $\mathbf{z_u}$, which means $\mathbf{z}_s$ fully reflects the class c but $\mathbf{z}_u$ dose not. Thus the objective can be rewritten as (derived in detail in Appendices).

$$\begin{aligned}\log p(\mathbf{x}) &= \log \iint \sum_c p(\mathbf{x}, \mathbf{z}_s, \mathbf{z}_u, c) d\mathbf{z}_s d\mathbf{z}_u \\ &\geq \mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u)] \\ &- D_{\mathrm{KL}}(q_\phi(\mathbf{z}_u|\mathbf{x})||p(\mathbf{z}_u)) \\ &- D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c))\end{aligned} \tag{2}$$

In Eq. 2, the ELBO becomes 3 terms in our setting. The **first** term is the negative reconstruction error, where p_θ is the decoder parameterized by θ. It measures whether the latent code $\mathbf{z}_s$ and $\mathbf{z}_u$ are informative enough to recover the original data. In practice, the reconstruction error L_{rec} can be defined as the l_2 loss between $\mathbf{x}$ and $\mathbf{x}'$. The **second** term acts as a regularization term of label irrelevant branch that pushes $q_\phi(\mathbf{z}_u|\mathbf{x})$ to match the prior distribution $p(\mathbf{z}_u)$, which is illustrated in detail in Section 3.2. The **third** term matches $q_\psi(\mathbf{z}_s|\mathbf{x})$ to a class-specific Gaussian distribution whose mean and covariance are learned with supervision, and it will be further introduced in Section 3.3.

3.2. Label irrelevant branch

Intuitively, we want to disentangle the latent code $\mathbf{z}$ into $\mathbf{z}_s$ and $\mathbf{z}_u$, and expect $\mathbf{z}_u$ to follow a fixed, prior distribution which is irrelevant to the label. This regularization is realized by minimizing KL divergence between $q_\phi(\mathbf{z}_u|\mathbf{x})$ and the prior $p(\mathbf{z}_u)$ as illustrated in Eq. 3. More specifically, $q_\phi(\mathbf{z}_u|\mathbf{x})$ is a Gaussian distribution whose mean $\boldsymbol{\mu}$ and diagonal covariance $\boldsymbol{\Sigma}$ are the output of $Encoder^u$ parameterized by ϕ. $p(\mathbf{z}_u)$ is simply set to $N(\mathbf{0}, \boldsymbol{I})$. Hence the KL regularization term is:

$$L_{kl} = D_{\mathrm{KL}}[\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(\mathbf{0}, \boldsymbol{I})] \tag{3}$$

Note that Eq. 3 can be represented in a closed form, which is easy to be computed.

To ensure good disentanglement in $\mathbf{z}_u$ and $\mathbf{z}_s$, we introduce adversarial learning in the latent space as in AAE [25] to drive the label relevant information out of $\mathbf{z}_u$. To do this, an adversarial classifier is added on the top of $\mathbf{z}_u$, which is trained to classify the category of $\mathbf{z}_u$ with cross entropy loss as is shown in (4):

$$L_C^{adv} = -\mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{x})} \sum_c \mathbb{I}(c = y) \log q_\omega(c|\mathbf{z}_u) \tag{4}$$

where $\mathbb{I}(c = y)$ is the indicator function, and $q_\omega(c|\mathbf{z}_u)$ is softmax probability output by the adversarial classifier parameterized by ω. Meanwhile, $Encoder^u$ is trained to fool the classifier, hence the target distribution becomes uniform

over all categories, which is $\frac{1}{C}$. The cross entropy loss is defined as (5).

$$L_E^{adv} = -\mathbb{E}_{q_\phi(\mathbf{z}_u|\mathbf{x})} \sum_c \frac{1}{C} \log q_\omega(c|\mathbf{z}_u) \tag{5}$$

3.3. Label relevant branch

Inspired by GM loss [44], we expect $\mathbf{z}_s$ to follow a Gaussian mixture distribution, expressed in Eq. 6, where $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ are the mean and covariance of Gaussian distribution for class c, and $p(c)$ is the prior probability, which is simply set to $\frac{1}{C}$ for all categories. For simplicity, we ignore the correlation among different dimensions of $\mathbf{z}_s$, hence $\boldsymbol{\Sigma}_c$ is assumed to be diagonal.

$$p(\mathbf{z}_s) = \sum_c p(\mathbf{z}_s|c)p(c) = \sum_c \mathcal{N}(\mathbf{z}_s; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)p(c) \tag{6}$$

Recall that in Eq. 2, the KL divergence between $q_\psi(\mathbf{z}_s, c|\mathbf{x})$ and $p(\mathbf{z}_s, c)$ is minimized. If $\mathbf{z}_s$ is formulated as a Gaussian distribution with its $\boldsymbol{\Sigma} \to \mathbf{0}$ and its mean $\hat{\mathbf{z}}_s$ output by $Encoder^s$, which is actually a Dirac delta function $\delta(\mathbf{z}_s - \hat{\mathbf{z}}_s)$, the KL divergence turns out to be the likelihood regularization term L_{lkd} in Eq. 7, which is proved in Appendices. Here $\boldsymbol{\mu}_y$ and $\boldsymbol{\Sigma}_y$ are the mean and covariance specified by the label y.

$$L_{lkd} = -\log \mathcal{N}(\hat{\mathbf{z}}_s; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) \tag{7}$$

Furthermore, we want $\mathbf{z}_s$ to contain label information as much as possible, thus the mutual information between $\mathbf{z}_s$ and class c is added to the maximization objective function. We prove in Appendices that it's equal to minimize the cross-entropy loss of the posterior probability $q(c|\mathbf{z}_s)$ and the label, which is exactly the classification loss L_{cls} in GM loss as is shown in Eq. 8.

$$\begin{aligned} L_{cls} &= -\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})} \sum_c \mathbb{I}(c = y) \log q(c|\mathbf{z}_s) \\ &= -\log \frac{\mathcal{N}(\hat{\mathbf{z}}_s|\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)p(y)}{\sum_k \mathcal{N}(\hat{\mathbf{z}}_s|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)p(k)} \end{aligned} \tag{8}$$

These two terms are added up to form GM loss in Eq. 9. Here L_{GM} is finally used to train the $Encoder^s$.

$$L_{GM} = L_{cls} + \lambda_{lkd} L_{lkd} \tag{9}$$

3.4. The decoder and the adversarial discriminator

The latent codes $\mathbf{z}_s$ and $\mathbf{z}_u$ output by $Encoder^s$ and $Encoder^u$ are first concatenated together, and then further given to the decoder to reconstruct the input $\mathbf{x}$ by $\mathbf{x}'$. Here the $Decoder$ is indicated by $p_\theta(\mathbf{x}|\mathbf{z})$ with its parameter θ learned from the l_2 reconstruction error L_{rec}. To synthesize a high quality $\mathbf{x}'$, we also employ the adversarial training in the pixel domain. Specifically, a discriminator $D_{\theta_d}(\mathbf{x}, c)$ with adversarial training on its parameter θ_d is used to improve $\mathbf{x}'$. Here the label c is utilized in D_{θ_d} like in [30]. The adversarial training loss for discriminator can be formulated as in Eq. 10,

$$\begin{aligned} L_D^{adv} = &- \mathbb{E}_{\mathbf{x}\sim P_r}[\log D_{\theta_d}(\mathbf{x}, c)] \\ &- \mathbb{E}_{\mathbf{z}_u\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\mathbf{z}_s\sim p(\mathbf{z}_s)}[\log(1 - D_{\theta_d}(G(\mathbf{z}_s, \mathbf{z}_u), c))] \end{aligned} \tag{10}$$

while this loss becomes

$$L_{GD}^{adv} = -\mathbb{E}_{\mathbf{z}_u\sim\mathcal{N}(\mathbf{0},\boldsymbol{I}),\mathbf{z}_s\sim p(\mathbf{z}_s)}[\log(D_{\theta_d}(G(\mathbf{z}_s, \mathbf{z}_u), c))]$$

for the generator. Note that here $G(\mathbf{z}_s, \mathbf{z}_u)$ is the decoder and $p(\mathbf{z}_s)$ is defined in Eq. 6.

3.5. Training algorithm

The training detail is illustrated in Algorithm 1. The $Encoder^s$, modeled by q_ψ, extracts label relevant code $\mathbf{z}_s$. $Encoder^s$ is trained with L_{GM} and L_{rec}, encouraging $\mathbf{z}_s$ to be label dependent and follow a learned Gaussian mixture distribution. Meanwhile, the $Encoder^u$ represented by q_ϕ is intended to extract class irrelevant code $\mathbf{z}_u$. It's trained by L_{kl}, L_E^{adv} and L_{rec} to make $\mathbf{z}_u$ irrelevant to the label and be close to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The adversarial classifier parameterized by ω is learned to classify $\mathbf{z}_u$ using L_C^{adv}. Then the decoder p_θ generates reconstruction image using the combined feature of $\mathbf{z}_s$ and $\mathbf{z}_u$ with the loss L_{rec}.

In the training process, a 2-stage alternating training algorithm is adopted. First, $Encoder^s$ is updated using L_{GM} to learn mean $\boldsymbol{\mu}_c$ and covariance $\boldsymbol{\Sigma}_c$ of the prior $p(\mathbf{z}_s|c)$. Then, the two encoders and the decoder are trained jointly to reconstruct images while the distributions of $\mathbf{z}_s$ and $\mathbf{z}_u$ are considered.

3.6. Application in semi-supervised generation

Given L unlabeled extra data $\mathcal{D}_u = \{\mathbf{x}^{(N+1)}, \mathbf{x}^{(N+2)}, \cdots, \mathbf{x}^{(N+L)}\}$, we now use our architecture for the semi-supervised generation, in which the labels $y^{(N+i)}$ of $\mathbf{x}^{(N+i)}$ in $\mathcal{D}_u$ are not presented. Here we hold the assumption that $\mathcal{D}_u$ are in the same domain as the fully supervised $\mathcal{D}_s$, but $y^{(N+i)}$ can be satisfied $y^{(N+i)} \in \{0, 1, \cdots, C-1\}$, or out of the predefined range. In other words, if the absent $y^{(N+i)}$ is in the predefined range, its $\mathbf{z}_s$ follows the same Gaussian mixture distribution as in Eq. 6. Otherwise, $\mathbf{z}_s$ should follow an ambiguous Gaussian distribution defined in Eq. 11.

$$\begin{aligned} \boldsymbol{\mu}_t &= \sum_c p(c)\boldsymbol{\mu}_c \\ \boldsymbol{\sigma}_t^2 &= \sum_c p(c)\boldsymbol{\sigma}_c^2 + \sum_c p(c)(\boldsymbol{\mu}_c)^2 - (\sum_c p(c)\boldsymbol{\mu}_c)^2 \end{aligned} \tag{11}$$

Algorithm 1 The training process of our proposed architecture.

Require: $\psi, \phi, \theta, \omega, \theta_d$ initial parameters of $Encoder^s$, $Encoder^u$, $Decoder$, the adversarial classifier on $\mathbf{z}_u$ and the discriminator on $\mathbf{x}$; $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}_c$ initial mean and covariance for Gaussian distribution of $\mathbf{z}_s$; n_{gm}, the number of iterations of L_{GM} per end-to-end iteration; λ_{rec} and λ_{kl} the weight of L_{rec} and L_{kl};

while not converged **do**
  for $i = 0$ to n_{gm} **do**
    Sample $\{\mathbf{x}, \mathrm{y}\}$ a batch from dataset.
    $\hat{\mathbf{z}}_s \leftarrow Encoder^s(\mathbf{x})$.
    $L_{GM} \leftarrow -\log q(y|\hat{\mathbf{z}}_s) - \lambda_{lkd} \log p(\hat{\mathbf{z}}_s|y)$
    $\psi \xleftarrow{+} -\nabla_\psi L_{GM}$
    $\boldsymbol{\mu}_c \xleftarrow{+} -\nabla_{\boldsymbol{\mu}_c} L_{GM}, c \in [0, C-1]$
    $\boldsymbol{\Sigma}_c \xleftarrow{+} -\nabla_{\boldsymbol{\Sigma}_c} L_{GM}, c \in [0, C-1]$
  end for
  Sample $\{\mathbf{x}, \mathrm{y}\}$ a batch from dataset.
  $\boldsymbol{\mu}, \boldsymbol{\Sigma} \leftarrow Encoder^u(\mathbf{x})$
  $L_{kl} \leftarrow D_{KL}[\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})||\mathcal{N}(\mathbf{0}, \boldsymbol{I})]$
  Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{z}_u \leftarrow \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\epsilon} + \boldsymbol{\mu}$
  $L_E^{adv} \leftarrow -\sum_c \frac{1}{C} \log q_\omega(c|\mathbf{z}_u)$
  $L_C^{adv} \leftarrow -\log q_\omega(y|\mathbf{z}_u)$
  $\hat{\mathbf{z}}_s \leftarrow Encoder^s(\mathbf{x})$.
  $L_{lkd} \leftarrow -\log \mathcal{N}(\hat{\mathbf{z}}_s; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$
  $\mathbf{x}'_f \leftarrow Decoder(\hat{\mathbf{z}}_s, \mathbf{z}_u)$
  $L_{rec} \leftarrow \frac{1}{2}||\mathbf{x} - \mathbf{x}'_f||_2^2$
  Sample $\mathbf{z}_s^p \sim p(\mathbf{z}_s|y),\ \mathbf{z}_u^p \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}'_p \leftarrow Decoder(\mathbf{z}_s^p, \mathbf{z}_u^p)$
  $L_D^{adv} \leftarrow -\log D_{\theta_d}(\mathbf{x}, y) - \log(1 - D_{\theta_d}(\mathbf{x}'_f, y)) - \log(1 - D_{\theta_d}(\mathbf{x}'_p, y))$
  $L_{GD}^{adv} \leftarrow -\log(D_{\theta_d}(\mathbf{x}'_f, y)) - \log(D_{\theta_d}(\mathbf{x}'_p, y))$
  $\psi \xleftarrow{+} -\nabla_\psi(L_{rec} + \lambda_{lkd} L_{lkd})$
  $\phi \xleftarrow{+} -\nabla_\phi(L_E^{adv} + \lambda_{kl} L_{kl} + \lambda_{rec} L_{rec})$
  $\omega \xleftarrow{+} -\nabla_\omega L_C^{adv}$
  $\theta \xleftarrow{+} -\nabla_\theta(L_{rec} + L_{GD}^{adv})$
  $\theta_d \xleftarrow{+} -\nabla_{\theta_d} L_D^{adv}$
end while

More specifically, $\mathbf{z}_s$ is expected to follow $\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$ where $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ are the total mean and covariance of all the class-specific Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ as illustrated in Eq. 6. Here, $\boldsymbol{\Sigma}_t$ is diagonal matrix with $\boldsymbol{\sigma}_t^2$ as its variance vector. $\boldsymbol{\sigma}_c^2$ is also the variance vector of $\boldsymbol{\Sigma}_c$. Hence, the likelihood regularization term becomes $L_{lkd} = -\log \mathcal{N}(\hat{\mathbf{z}_s}; \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$. The whole network is trained in a end-to-end manner using total losses. Note that in this setting, the label y is not provided, so L_{GM} , L_E^{adv} and L_C^{adv} are ignored in the training process.

4. Experiments

In this section, experiments are carried out to validate the effectiveness of the proposed method. A toy example is first designed to show that by disentangling the label relevant and irrelevant codes, our model has the ability of generating diverse data samples than cVAE-GAN [3]. We then compare the quality of generated images on real image datasets. The latent space is also analyzed. Finally, the experiments of semi-supervised generation and image inpainting show the flexibility of our model, hence it may have many potential applications.

4.1. Toy examples

This section demonstrates our method on a toy example, in which the real data distribution lies in 2D with one dimension (x axis) being label relevant and the other (y axis) being irrelevant. The distribution is assumed to be known. There are 3 types of data points indicated by green, red and blue, belonging to 3 classes. The 2D data points and their corresponding labels are given to our model for variational inference and the new sample generation.

For comparison, we also give the same training data to cVAE-GAN for the same purpose. The two compared models share the similar settings of the network. In our model, the two encoders are both MLP with 3 hidden layers, and there are 32, 64, and 64 units in them. In cVAE-GAN, the encoder is the same, but it only has one encoder. The discriminators are exactly the same, which is also an MLP of 3 hidden layers with 32, 64, and 64 units. Adam is used as the optimization method in which a fixed learning rate of 0.0005 is applied for both. Each model is trained for 50 epochs until they all converge. The generated samples of each model are plotted in Figure 2.

From Figure 2 we can observe that both two models can capture the underlying data distribution, and our model converges at the similar rate. The advantage of our model is that it tends to generate diverse samples, while cVAE-GAN generates samples in a conserving way in which the label irrelevant dimensions are within the limited value range.

4.2. Analysis on generated image quality

In this section, we compare our method with other generative models for image generation quality. The experiments are conducted on two datasets: FaceScrub [31] and CIFAR-10 [21]. The FaceScrub contains $92k$ training images from 530 different identities. For FaceScrub, a cascaded object detector proposed in [42] is first used to detect faces first, and then the face alignment is also conducted based on SDM proposed in [46]. The detected cropped faces are resized to the fixed size 64×64. In the training process, Adam optimizer with $\alpha = 0.0005$ is used. The hyper parameter λ_{lkd}, λ_{kl}, and λ_{rec} are set to 0.1, $\frac{10}{N_{pixel}}$, and $\frac{1}{N_{\mathbf{z}_u}}$,

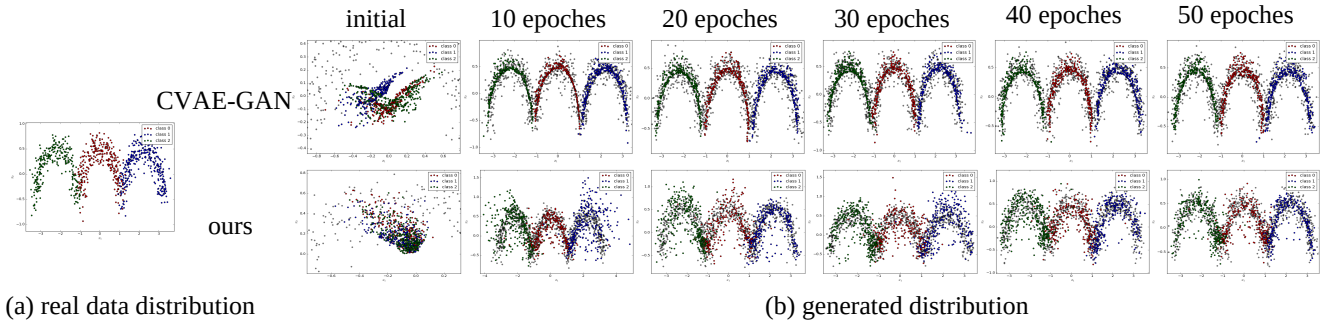


Figure 2. Results on a toy example for our model and cVAE-GAN. We show the generated points at different epochs.

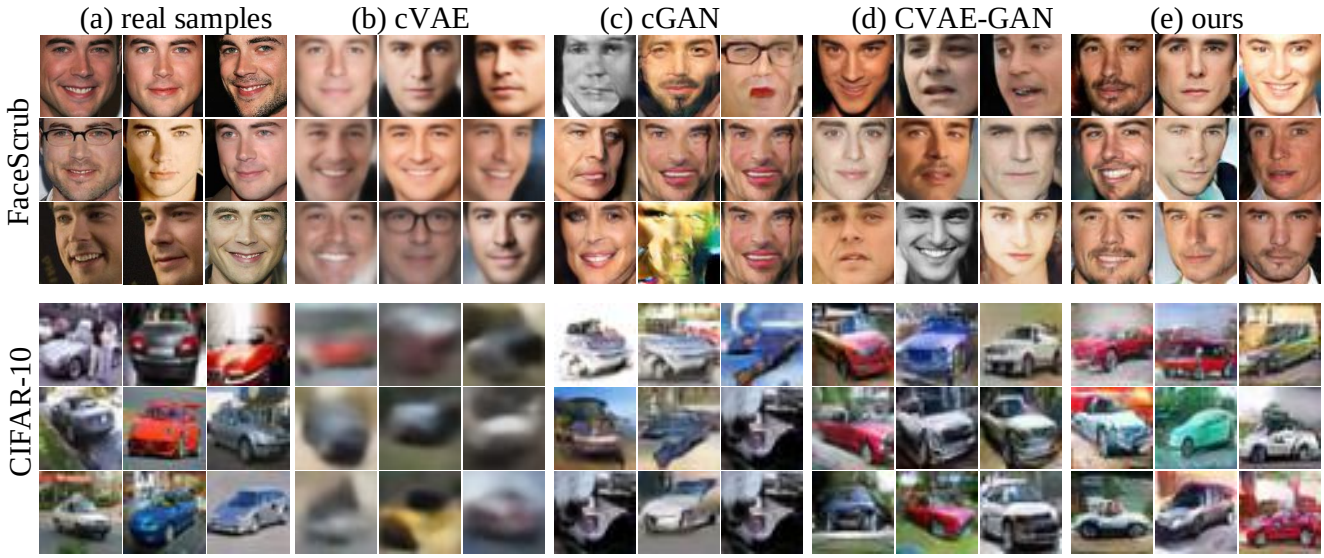


Figure 3. Visualization of generated images of different models.

respectively. Here, N_{pixel} is the number of image pixels, and $N_{\mathbf{z}_u}$ is the dimension of $\mathbf{z}_u$. Since our method incorporates the label for training, popular generative networks conditioned on label, like cVAE [39], cVAE-GAN [3], and cGAN [30], are chosen for comparison. For cVAE, cVAE-GAN and cGAN, we randomly generate samples of class c by first sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then concatenating $\mathbf{z}$ and one hot vector of c as the input of decoder/generator. As for ours, $\mathbf{z}_s \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)$ and $\mathbf{z}_u \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are sampled and combined for decoder to generate samples. Some of generated images are visualized in Figure 9. It shows that samples generated by cVAE are highly blurred, and cGAN suffers from mode collapse. Samples generated by cVAE-GAN and our method seem to have similar quality, we refer to two metrics, $Inception\ Score$ [36] and intra-class diversity [5] to compare them.

We adopt $Inception\ Score$ to evaluate realism and inter-class diversity of images. Generated images that are close to real images of class y should have a posterior probability $p(y|\mathbf{x})$ with low entropy. Meanwhile, images of diverse classes should have a marginal probability $p(y)$ with high entropy. Hence, $Inception\ Score$, formulated as $\exp(\mathbb{E}_{\mathbf{x}} KL(p(y|\mathbf{x})||p(y)))$, gets a high value when images are realistic and diverse.

To get conditional class probability $p(y|\mathbf{x})$, we first train a classifier with Inception-ResNet-v1 [40] architecture on real data. Then we randomly generate 53k samples(100 for each class) of FaceScrub and 5k samples (500 for each class) of CIFAR-10, and apply them to the pre-trained classifier. The marginal $p(y)$ is obtained by averaging all $p(y|\mathbf{x})$. The results are listed in Table 4.

We emphasize that our method will generate more diverse samples in one class. Since $Inception\ Score$ only measures inter-class diversity, intra-class diversity of samples should also be taken into account. We adopt the metric proposed in [5], which measures the average negative MS-SSIM [45] between all pairs in the generated image set $\boldsymbol{X}$. Table 2 shows the inter-class diversity of cVAE-GAN and

	FaceScrub	CIFAR-10
cVAE [39]	9.55	3.01
cGAN [30]	10.02	6.27
cVAE-GAN [3]	16.75	6.99
ours	**17.91**	**7.04**

Table 1. $Inception\ Score$ of different methods on two datasets. Please refer to 4.2 for more details.

	FaceScrub	CIFAR-10
cVAE-GAN [3]	0.0141	0.0136
ours	**0.0157**	**0.0149**

Table 2. Intra-class diversity of different methods on two datasets. Please refer to 4.2 for more details.

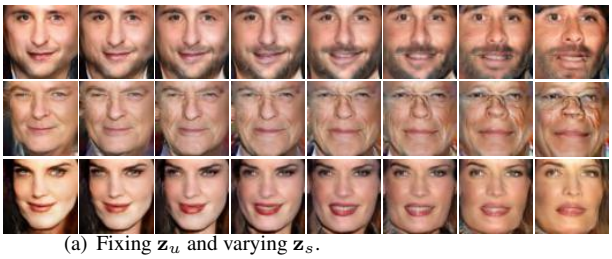
our method on FaceScrub and CIFAR-10.

$$d_{intra}(\mathbf{X}) = 1 - \frac{1}{|\mathbf{X}|^2} \sum_{(\mathbf{x}',\mathbf{x})\in\mathbf{X}\times\mathbf{X}} MS-SSIM(\mathbf{x}',\mathbf{x}) \tag{12}$$

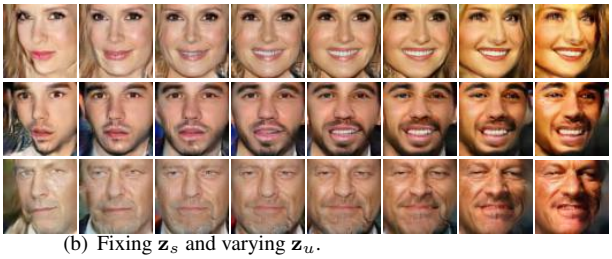
4.3. Analysis on disentangled latent space

We now evaluate our proposal on the disentangled latent space, which is represented by label relevant dimensions $\mathbf{z}_s$ and irrelevant ones $\mathbf{z}_u$. $\mathbf{z}_s$ for class c is supposed to capture the variation unique to training images within the label c, while $\mathbf{z}_u$ should contain the variation in common characteristics for all classes. It's validated in the following ways: (1) fixing $\mathbf{z}_u$ and varying $\mathbf{z}_s$. In this setting, we directly sample a $\mathbf{z}_u \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and keep it fixed. Then a set of $\mathbf{z}_s$ for class c is obtained by first getting a series of random codes sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and then mapping them to class c. In specific, we first sample $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\mathbf{z}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then a set of random codes $\mathbf{z}^{(i)}$ are obtained by linear interpolation, i.e., $\mathbf{z}^{(i)} = \alpha\mathbf{z}_1 + (1-\alpha)\mathbf{z}_2, \alpha \in [0, 1]$. We map each $\mathbf{z}^{(i)}$ to class c with $\mathbf{z}_s^{(i)} = \mathbf{z}^{(i)} \odot \boldsymbol{\sigma}_c + \boldsymbol{\mu}_c$. Finally each $\mathbf{z}_s^{(i)}$ is concatenated with the fixed $\mathbf{z}_u$ and given to the decoder to get a generated image. (2) fixing $\mathbf{z}_s$ and varying $\mathbf{z}_u$. Similar to (1), we first sample a $\mathbf{z}_s \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{c}, \boldsymbol{\sigma}_c)$ from a learned distribution and keep it fixed. Then a set of label irrelevant $\mathbf{z}_u$ are obtained by linearly interpolating between $\mathbf{z}_1$ and $\mathbf{z}_2$, where $\mathbf{z}_1$ and $\mathbf{z}_2$ are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

We conduct experiments on FaceScrub and the generated images are shown in Figure 4. In Figure 4 (a), each row presents samples generated by linearly transformed $\mathbf{z}_s$ of a certain class c and a fixed $\mathbf{z}_u$. All three rows share the same $\mathbf{z}_u$, and each column shares the same random code $\mathbf{z}^{(i)}$ and just maps it to different class c with $\mathbf{z}_s^{(i)} = \mathbf{z}^{(i)} \odot \boldsymbol{\sigma}_c + \boldsymbol{\mu}_c$. It shows that as $\mathbf{z}_s$ varies, one may change differently for different identities, e.g., grow a beard, wrinkle, or take off the make-up. In Figure 4 (b), each row presents samples with linearly transformed $\mathbf{z}_u$ a fixed $\mathbf{z}_s$ of class c, and each column shares a same $\mathbf{z}_u$. We can see that images from each row change consistently with poses, expressions and illuminations. These two experiments suggest that $\mathbf{z}_s$ is relevant to c, while $\mathbf{z}_u$ reflects more common label irrelevant characteristics.



(a) Fixing $\mathbf{z}_u$ and varying $\mathbf{z}_s$.



(b) Fixing $\mathbf{z}_s$ and varying $\mathbf{z}_u$.

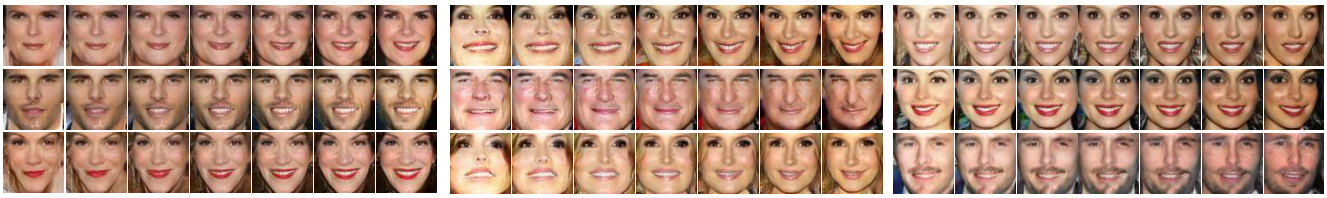
Figure 4. The generated images by fixing one code and varying the other. In (a), each row shows samples for linearly transformed $\mathbf{z}_s$ of a certain class c with a fixed $\mathbf{z}_u$. In (b), each row corresponds to samples for linearly transformed $\mathbf{z}_u$ with a fixed $\mathbf{z}_s$ of class c.

We are also interested in each dimension in $\mathbf{z}_u$ and conduct an experiment by varying a single element in it. We find three dimensions in $\mathbf{z}_u$ which reflect the meaningful the common characteristics, such as the expression, elevation and azimuth.

4.4. Semi-supervised image generation

According to the details in Section 3.6, the experiments on semi-supervised image generation are conducted. We find our method can learn well disentangled latent representation when the unlabeled extra data are available. To validate that, we randomly select 200 identities of about 21k images from CASIA [47] dataset and remove their labels to form unlabeled dataset $\mathcal{D}_u$. Note that the identities in $\mathcal{D}_u$ are totally different with those in FaceScrub. After training the whole network on labeled dataset $\mathcal{D}_s$, we finetune it on $\mathcal{D}_u$ using the training algorithm illustrated in Section 3.6.

To demonstrate the semi-supervised generation results, two different images are given to $Encoder^S$ and $Encoder^U$ to generate the code $\mathbf{z}_s$ and $\mathbf{z}_u$, respectively. Then, the decoder is required to synthesis a new image based on the concatenated code from $\mathbf{z}_s$ and $\mathbf{z}_u$. The Figure 6 shows face synthesis results using images whose identities have not appeared in $\mathcal{D}_s$. The first row and first column show a set of original images providing $\mathbf{z}_u$ and $\mathbf{z}_s$ respectively,



(a) expression (b) elevation (c) azimuth

Figure 5. The generated images by fixing $\mathbf{z}_s$ for each row and varying single dimensions in $\mathbf{z}_u$. Here, we find three different dimensions in $\mathbf{z}_u$, which directly causes the variations on expressions in (a), elevation in (b), and azimuth in (c).

Figure 6. Face synthesis using images whose identities have not appeared in $\mathcal{D}_s$. Original images providing $\mathbf{z}_u$ and $\mathbf{z}_s$ are given in the first row and the first column. The synthesizing images using the combination of $\mathbf{z}_u$ $\mathbf{z}_s$ are shown in the corresponding position.

while images in the middle are generated ones using $\mathbf{z}_s$ of the corresponding row and $\mathbf{z}_u$ of the corresponding column. It is obvious that the identity depends on $\mathbf{z}_s$, while other characteristics like the poses, illumination, expressions are reflected on $\mathbf{z}_u$. This semi-supervised generation shows $\mathbf{z}_s$ and $\mathbf{z}_u$ can also be disentangled on identities outside the labeled training data $\mathcal{D}_s$, which provides the great flexibility for image generation.

4.5. Image inpainting

Our method can also be applied to image inpainting. It means that given a partly corrupted image, we can extract meaningful latent code to reconstruct the original image. Note that in cVAE-GAN [3], an extra class label c should be provided for reconstruction while it's needless in our method. In practice, we first corrupt some patches for a image $\boldsymbol{x}$, namely right-half, eyes, nose and mouth, and bottom-half regions, then input those corrupted images into the two encoders to get $\mathbf{z}_s$ and $\mathbf{z}_u$, then the reconstructed image $\boldsymbol{x'}$ is generated using a combined $\mathbf{z}_s$ and $\mathbf{z}_u$. The image inpainting result is obtained by $\boldsymbol{x}^{inp} = \boldsymbol{M} \odot \boldsymbol{x'} + (1 - \boldsymbol{M}) \odot \boldsymbol{x}$, where $\boldsymbol{M}$ is the binary mask for the corrupted patch. Figure 7 shows the results of image inpainting. cVAE-GAN struggles to complete the images when it comes to a large part of missing regions (e.g. right-half and bottom-half parts) or pivotal regions of faces (e.g. eyes), while our method provides visually pleasing results.

Figure 7. Image inpainting results. The original image is corrupted with different patterns, on the right-half, eyes, nose and mouth, and bottom-half face. We compare our model with cVAE-GAN.

5. Conclusion

We propose a latent space disentangling algorithm on VAE baseline. Our model learns two separated encoders

and divides the latent code into label relevant and irrelevant dimensions. Together with a discriminator in pixel domain, we show that our model can generate high quality and diverse images, and it can also be applied in semi-supervised image generation in which unlabeled data with unseen classes are given to the encoders. Future research includes building more interpretable latent dimensions with help of more labels, and reducing the correlation between the label relevant and irrelevant codes in our framework.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Project 61302125, and in part by Natural Science Foundation of Shanghai under Project 17ZR1408500. Corresponding to sunli@ee.ecnu.edu.cn

References

[1] M. E. Abbasnejad, A. Dick, and A. van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 781–790. IEEE, 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *stat*, 1050:9, 2017.

[3] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. Cvae-gan: Fine-grained image generation through asymmetric training. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2764–2773. IEEE, 2017.

[4] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua. Towards open-set identity preserving face synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6713–6722, 2018.

[5] M. Ben-Yosef and D. Weinshall. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *arXiv preprint arXiv:1808.10356*, 2018.

[6] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[7] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[8] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *Automatic Face & Gesture Recognition (FG 2018), 2018 13th IEEE International Conference on*, pages 67–74. IEEE, 2018.

[9] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[10] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[11] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *ICLR*, 2017.

[12] Y. Ge, Z. Li, H. Zhao, G. Yin, X. Wang, and H. Li. Fd-gan: Pose-guided feature distilling gan for robust person re-identification. In *Advances in Neural Information Processing Systems*, 2018.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[15] N. Hadad, L. Wolf, and M. Shahar. A two-step disentanglement method. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 772–780, 2018.

[16] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

[17] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[18] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[19] Y. Kim, S. Wiseman, A. C. Miller, D. Sontag, and A. M. Rush. Semi-amortized variational autoencoders. *arXiv preprint arXiv:1802.02550*, 2018.

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *stat*, 1050:1, 2014.

[21] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[22] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pages 1558–1566, 2016.

[23] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. K. Singh, and M.-H. Yang. Diverse image-to-image translation via disentangled representations. In *European Conference on Computer Vision*, 2018.

[24] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[25] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[26] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2813–2821. IEEE, 2017.

[27] M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation

in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

[28] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning (ICML)*, pages 2391–2400. PMLR, 2017.

[29] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[30] T. Miyato and M. Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

[31] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014.

[32] S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[33] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *International Conference on Machine Learning*, pages 2642–2651, 2017.

[34] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[36] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

[37] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. Neural face editing with intrinsic image disentangling. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5444–5453. IEEE, 2017.

[38] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*, pages 3483–3491, 2015.

[40] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.

[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[42] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[43] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[44] W. Wan, Y. Zhong, T. Li, and J. Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9117–9126, 2018.

[45] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[46] X. Xiong and F. De la Torre. Supervised descent method and its applications to face alignment. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 532–539, 2013.

[47] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014.

Appendices

A. Mathematical proofs

A.1. The ELBO of the log-likelihood objective

. We declare in Equation 2 that after dividing the latent space into label relevant dimensions $\mathbf{z}_s$ and label irrelevant dimensions $\mathbf{z}_u$, the ELBO of the log-likelihood objective $\log p(\mathbf{x})$ becomes 3 terms in our setting.

$$\begin{aligned}\log p(\mathbf{x}) &= \log \iint p(\mathbf{x}, \mathbf{z}_s, \mathbf{z}_u) d\mathbf{z}_s d\mathbf{z}_u \\ &\geq \mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u)] \\ &- D_{\mathrm{KL}}(q_\phi(\mathbf{z}_u|\mathbf{x})||p(\mathbf{z}_u)) \\ &- D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c))\end{aligned}$$

Proof

Our generative process is described as follows. First, sample a label relevant code $\mathbf{z}_s \sim p(\mathbf{z}_s, c)$ and a label irrelevant code $\mathbf{z}_u \sim p(\mathbf{z}_u)$. Then, a decoder $p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u)$, taking the combination of $\mathbf{z}_s$ and $\mathbf{z}_u$ as input, maps latent codes to images. Hence, we factorize the joint distribution $p(\mathbf{x}, \mathbf{z}_s, \mathbf{z}_u)$ as:

$$p(\mathbf{x}, \mathbf{z}_s, \mathbf{z}_u) = \sum_c p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u) p(\mathbf{z}_s, c) p(\mathbf{z}_u)$$

By using Jensens inequality, the log-likelihood $\log p(\mathbf{x})$ can be written as:

$$\begin{aligned}\log p(\mathbf{x}) &= \log \iint p(\mathbf{x}, \mathbf{z}_s, \mathbf{z}_u) d\mathbf{z}_s d\mathbf{z}_u \\ &= \log \iint \sum_c p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u) p(\mathbf{z}_s, c) p(\mathbf{z}_u) d\mathbf{z}_s d\mathbf{z}_u \\ &= \log \mathbb{E}_{q_\psi(\mathbf{z}_s, c|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})} \frac{p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u) p(\mathbf{z}_u) p(\mathbf{z}_s, c)}{q_\psi(\mathbf{z}_s, c|\mathbf{x}) q_\phi(\mathbf{z}_u|\mathbf{x})} \\ &\geq \mathbb{E}_{q_\psi(\mathbf{z}_s, c|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log \frac{p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u) p(\mathbf{z}_u) p(\mathbf{z}_s, c)}{q_\psi(\mathbf{z}_s, c|\mathbf{x}) q_\phi(\mathbf{z}_u|\mathbf{x})}] \\ &= \mathbb{E}_{q_\psi(\mathbf{z}_s, c|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u)] \\ &\quad + \mathbb{E}_{q_\psi(\mathbf{z}_s, c|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log \frac{p(\mathbf{z}_u)}{q_\phi(\mathbf{z}_u|\mathbf{x})}] \\ &\quad + \mathbb{E}_{q_\psi(\mathbf{z}_s, c|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log \frac{p(\mathbf{z}_s, c)}{q_\psi(\mathbf{z}_s, c|\mathbf{x})}] \\ &= \mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x}), q_\phi(\mathbf{z}_u|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}_s, \mathbf{z}_u)] \\ &\quad - D_{\mathrm{KL}}(q_\phi(\mathbf{z}_u|\mathbf{x})||p(\mathbf{z}_u)) \\ &\quad - D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c))\end{aligned}$$

A.2. Log-likelihood regularization term in the label relevant branch

Note that the KL divergence $D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c))$, the third term of the ELBO in Equation 2, is minimized. If we assume conditional independence between $\mathbf{z}_s$ and the class c, then we have

$$q_\psi(\mathbf{z}_s, c|\mathbf{x}) = q_\psi(\mathbf{z}_s|\mathbf{x}) p(c|x)$$

where $p(c|x)$ is the one-hot encoding of the label y. If $q_\psi(\mathbf{z}_s|\mathbf{x})$ is formulated as Gaussian distribution with $\mathbf{\Sigma} \to \mathbf{0}$ and mean $\hat{\mathbf{z}}_s$ output by $Encoder^s$, which is actually a Dirac delta function.

$$q_\psi(\mathbf{z}_s|\mathbf{x}) = \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s)$$

The KL regularization term becomes

$$\begin{aligned}&D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c)) \\ =&D_{\mathrm{KL}}[\delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x})||p(\mathbf{z}_s|c) p(c)] \\ =&-\sum_c \int \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x}) \log \frac{p(\mathbf{z}_s|c) p(c)}{\delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x})} d\mathbf{z}_s \\ =&-\sum_c \int \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x}) \log p(\mathbf{z}_s|c) d\mathbf{z}_s \\ &-\sum_c \int \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x}) \log \frac{p(c)}{p(c|\mathbf{x})} d\mathbf{z}_s \\ &+\sum_c \int \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) p(c|\mathbf{x}) \log \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) d\mathbf{z}_s \\ =&-\sum_c p(c|\mathbf{x}) \log p(\hat{\mathbf{z}}_s|c) - \sum_c p(c|\mathbf{x}) \log \frac{p(c)}{p(c|\mathbf{x})} \\ &+\int \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) \log \delta(\mathbf{z}_s - \hat{\mathbf{z}}_s) d\mathbf{z}_s\end{aligned}$$

The second term relates to the prior distribution, so it can be regraded as a constant. The third term is negative entropy of delta function and has nothing to do with $\hat{\mathbf{z}}_s$, hence we consider it as a constant too. Therefore, we have

$$\begin{aligned}&D_{\mathrm{KL}}(q_\psi(\mathbf{z}_s, c|\mathbf{x})||p(\mathbf{z}_s, c)) \\ =&-\sum_c p(c|\mathbf{x}) \log p(\hat{\mathbf{z}}_s|c) + Const. \\ =&-\sum_c \mathbb{I}(c = y) \log p(\hat{\mathbf{z}}_s|c) + Const.\end{aligned}$$

where the prior distribution $p(\hat{\mathbf{z}}_s|c)$ is set to $N(\hat{\mathbf{z}}_s; \boldsymbol{\mu}_c, \mathbf{\Sigma}_c)$. Ignoring the constant term, it turns out to be the likelihood regularization term L_{lkd} in Equation 7.

$$\begin{aligned}L_{lkd} &= -\sum_c \mathbb{I}(c = y) \log N(\hat{\mathbf{z}}_s; \boldsymbol{\mu}_c, \mathbf{\Sigma}_c) \\ &= -\log N(\hat{\mathbf{z}}_s; \boldsymbol{\mu}_y, \mathbf{\Sigma}_y)\end{aligned}$$

A.3. Cross-entropy objective in the label relevant branch

To encourage $\mathbf{z}_s$ to become label relevant as much as possible, the mutual information $I(\mathbf{z}_s; c)$ is maximized,

where $\mathbf{z}_s \sim q_\psi(\mathbf{z}_s|\mathbf{x})$. In practice, $I(\mathbf{z}_s; c)$ is hard to optimize directly because it requires access to $p(c|\mathbf{z}_s)$. We can instead optimize its lower bound by introducing an auxiliary distribution $q(c|\mathbf{z}_s)$ to approximate $p(c|\mathbf{z}_s)$ as in infoGAN [9] .

$$\begin{aligned} I(\mathbf{z}_s; c) &= H(c) - H(c|\mathbf{z}_s) \\ &= H(c) + \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log p(c|\mathbf{z}_s) \\ &= H(c) + \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log \frac{p(c|\mathbf{z}_s)}{q(c|\mathbf{z}_s)} \\ &+ \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log q(c|\mathbf{z}_s) \\ &\geq H(c) + \mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log q(c|\mathbf{z}_s) \end{aligned}$$

Since we still need to sample from $p(c|\mathbf{z}_s)$ in the inner expectation, we adopt Lemma 5.1 in infoGAN to further remove the need of $p(c|\mathbf{z}_s)$. The first term of the lower bound is a constant, so we ignore it. Then the second term becomes

$$\begin{aligned} &\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log q(c|\mathbf{z}_s) \\ =&\mathbb{E}_{p(c')}\mathbb{E}_{p(\mathbf{x}|c')}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\mathbb{E}_{p(c|\mathbf{z}_s)} \log q(c|\mathbf{z}_s) \\ =&\sum_{c'}\sum_{c}\iint p(c')p(\mathbf{x}|c')q_\psi(\mathbf{z}_s|\mathbf{x})p(c|\mathbf{z}_s) \log q(c|\mathbf{z}_s) d\mathbf{x} d\mathbf{z}_s \\ =&\sum_{c'}\sum_{c}\int p(c')p(c|\mathbf{z}_s) \log q(c|\mathbf{z}_s)[\int p(\mathbf{x}|c')q_\psi(\mathbf{z}_s|\mathbf{x})d\mathbf{x}]d\mathbf{z}_s \end{aligned}$$

We hold the assumption that the process of sampling $\mathbf{z}_s|\mathbf{x}$ is independent on c, thus

$$\int p(\mathbf{x}|c')q_\psi(\mathbf{z}_s|\mathbf{x})d\mathbf{x} = \int p(\mathbf{z}_s, \mathbf{x}|c')d\mathbf{x} = p(\mathbf{z}_s|c')$$

According to Lemma 5.1 in infoGAN, we have

$$\begin{aligned} &\sum_{c'}\sum_{c}\int p(c')p(\mathbf{z}_s|c')p(c|\mathbf{z}_s) \log q(c|\mathbf{z}_s) d\mathbf{z}_s \\ =&\sum_{c'}\int p(c')p(\mathbf{z}_s|c') \log q(c'|\mathbf{z}_s) d\mathbf{z}_s \end{aligned}$$

Hence

$$\begin{aligned} &\sum_{c'}\sum_{c}\int p(c')p(c|\mathbf{z}_s) \log q(c|\mathbf{z}_s)[\int p(\mathbf{x}|c')q_\psi(\mathbf{z}_s|\mathbf{x})d\mathbf{x}]d\mathbf{z}_s \\ =&\sum_{c'}\sum_{c}\int p(c')p(\mathbf{z}_s|c')p(c|\mathbf{z}_s) \log q(c|\mathbf{z}_s) d\mathbf{z}_s \\ =&\sum_{c}\int p(c)p(\mathbf{z}_s|c) \log q(c|\mathbf{z}_s) d\mathbf{z}_s \end{aligned}$$

We further factorize $p(\mathbf{z}_s|c)$ as $\int p(\mathbf{x}|c)q_\psi(\mathbf{z}_s|\mathbf{x})d\mathbf{x}$, the equation above becomes

$$\begin{aligned} &\sum_{c}\int p(c)p(\mathbf{z}_s|c) \log q(c|\mathbf{z}_s) d\mathbf{z}_s \\ =&\sum_{c}\iint p(c)p(\mathbf{x}|c)q_\psi(\mathbf{z}_s|\mathbf{x}) \log q(c|\mathbf{z}_s) d\mathbf{z}_s d\mathbf{x} \\ =&\sum_{c}\iint p(\mathbf{x})q_\psi(\mathbf{z}_s|\mathbf{x})p(c|\mathbf{x}) \log q(c|\mathbf{z}_s) d\mathbf{z}_s d\mathbf{x} \\ =&\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\sum_{c} p(c|\mathbf{x}) \log q(c|\mathbf{z}_s) \end{aligned}$$

where $p(c|\mathbf{x})$ is the one-hot encoding of the label y, i.e. $p(c|\mathbf{x}) = \mathbb{I}(c = y)$. To maximize it is to minimize its opposite, which is exactly the classification loss in Section 3.3.

$$L_{cls} = -\mathbb{E}_{p(\mathbf{x})}\mathbb{E}_{q_\psi(\mathbf{z}_s|\mathbf{x})}\sum_{c} \mathbb{I}(c = y) \log q(c|\mathbf{z}_s)$$

B. Experimental details

B.1. Dataset synthesis of toy example

Our synthetic dataset of toy example is a modification of the two-moon dataset, which contains three half circles instead of two. The generative process is described as follows. First, sample data points from three half unit circles with a horizontal interval of 2.2. Then, add Gaussian noises with $std = 0.15$ to all of them.

B.2. Network architecture of FaceScrub

For the two encoders, $Encoder^s$ and $Encoder^u$, we use VGG [38] architecture with batch normalization layers added to each layer and replace the last three fc layers with two fc layers of 1024 and 512 units. For the decoder, an inverse structure of the encoders is applied. The adversarial classifier in Section 3.2 consists of two fc layers of 256 and 530 units, and the discriminator contains 7 convolution layers and two fc layers (details are shown in Table 4). Note that spectral normalization [29] is applied to to the all of the weights in the discriminator and the label embedding is incorporated in the first fc layer as in [30].

B.3. Network architecture of Cifar-10

The network structures of the two encoders, decoder and discriminator for Cifar-10 are shown in Table 3. The adversarial classifier in the latent space is similar as that used for FaceScrub, which are two fc layers of 256 and 10 units. Also, spectral normalization and label embedding are applied in the discriminator.

B.4. Optimization

We use Adam optimizer with $\alpha = 0.0005$, $\beta_1 = 0$ and $\beta_2 = 0.9$. Since in the training process, the first stage using

Encoder	Decoder	Discriminator
input $\mathbf{x} \in \mathbb{R}^{32\times32\times3}$	input $\mathbf{z}_s \in \mathbb{R}^{100}$, $\mathbf{z}_u \in \mathbb{R}^{200}$	input $\mathbf{x} \in \mathbb{R}^{32\times32\times3}$
5×5 conv, 32, stride 2, batchnorm, relu	concat	5×5 conv, 32, stride 1, lrelu
5×5 conv, 64, stride 2, batchnorm, relu	fc, 1024, batchnorm, relu	5×5 conv, 128, stride 2, lrelu
3×3 conv, 128, stride 2, batchnorm, relu	5×5 conv, 256, stride 2, batchnorm, relu	5×5 conv, 256, stride 2, lrelu
3×3 conv, 256, stride 2, batchnorm, relu	5×5 conv, 256, stride 1, batchnorm, relu	5×5 conv, 256, stride 2, lrelu
fc, 1024, batchnorm, relu	5×5 conv, 128, stride 2, batchnorm, relu	fc, 512, lrelu
fc, 100 (for $\mathbf{z}_s$) / 200 (for $\mathbf{z}_u$)	5×5 conv, 64, stride 2, batchnorm, relu	fc, 1
	5×5 conv, 32, stride 2, batchnorm, relu	
	5×5 conv, 3, stride 1, tanh	

Table 3. The network structure for Cifar-10.

Discriminator for FaceScrub
input $\mathbf{x} \in \mathbb{R}^{64\times64\times3}$
3×3 conv, 64, stride 2, lrelu
3×3 conv, 128, stride 2, lrelu
3×3 conv, 256, stride 1, lrelu
3×3 conv, 256, stride 2, lrelu
3×3 conv, 512, stride 1, lrelu
3×3 conv, 512, stride 2, lrelu
3×3 conv, 512, stride 2, lrelu
global average pooling
fc, 1024, lrelu
fc, 1

Table 4. The network structure of discriminator for FaceScrub.

L_{GM} is trained 3 times per second stage iteration, L_{GM} converges fast. Continuously training after it converges will cause instability of L_{GM} because $\mathbf{\Sigma}_c$ goes down gradually. In practice, we decay the learning rate of $\mathbf{\Sigma}_c$ by 0.01 after 2 epochs.

B.5. Inception Score

Recall that $Inception\ Score$ requires access to the conditional class probability $p(y|\mathbf{x})$. We use classification model of Inception-ResNet-v1 [40] architecture trained on VGGFace2 [8] to evaluate generative models trained on FaceScrub. For generative models trained on Cifar-10, classification model of Inception-v3 [41] architecture trained on ImageNet [35] is used.

C. Additional experiment results

C.1. More generated samples on FaceScrub and Cifar-10

Figure 8 shows generated samples of our method on FaceScrub and Cifar-10 with each row corresponding to a certain class.

C.2. Additional experiments on CUB-200-2011 and Cifar-100

We additionally apply our method to CUB-200-2011 [43] and Cifar-100 [21] dataset. The CUB-200-2011 contains 200 categories of birds with 11,788 images in total. For CUB-200-2011, we crop the images according to the bounding boxes provided by the dataset and resize the cropped images to 64×64. The network structure is just same as it used in FaceScrub. For Cifar-100, we use the same network as in Cifar-10. Generated images are shown in Figure 9. Results of $Inception\ Score$ and intra-class diversity are listed in Table 5 and Table 6, respectively.

	CUB-200-2011	Cifar-100
cVAE [39]	37.34	3.10
cGAN [30]	78.06	6.39
cVAE-GAN [3]	91.14	6.68
ours	**100.86**	**6.70**

Table 5. $Inception\ Score$ of different methods.

	CUB-200-2011	Cifar-100
cVAE-GAN [3]	**0.0195**	0.0179
ours	0.0192	**0.0190**

Table 6. Intra-class diversity of different methods.

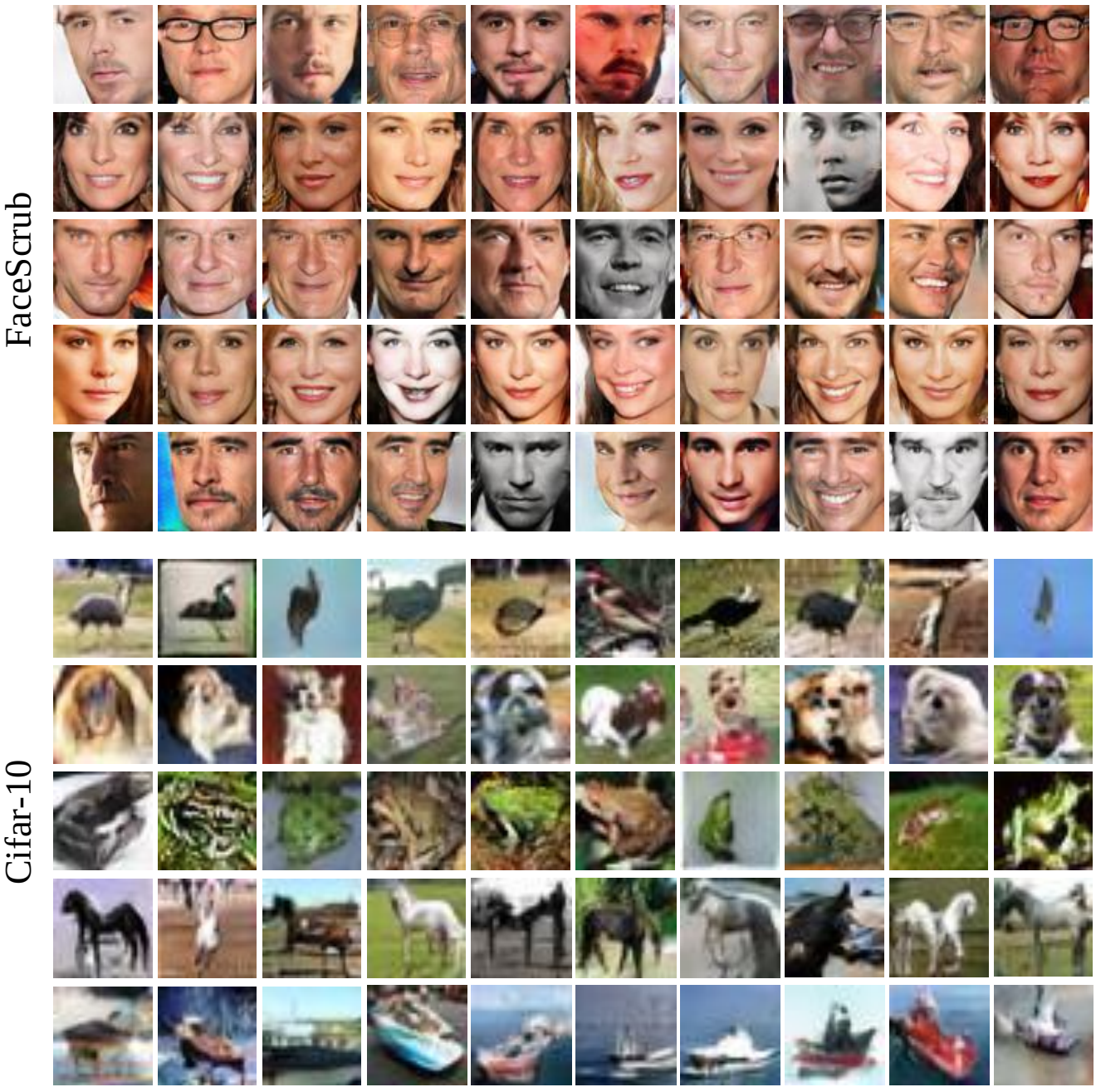


Figure 8. Generated samples of our method on FaceScrub and Cifar-10. Each row shows images of a certain class.

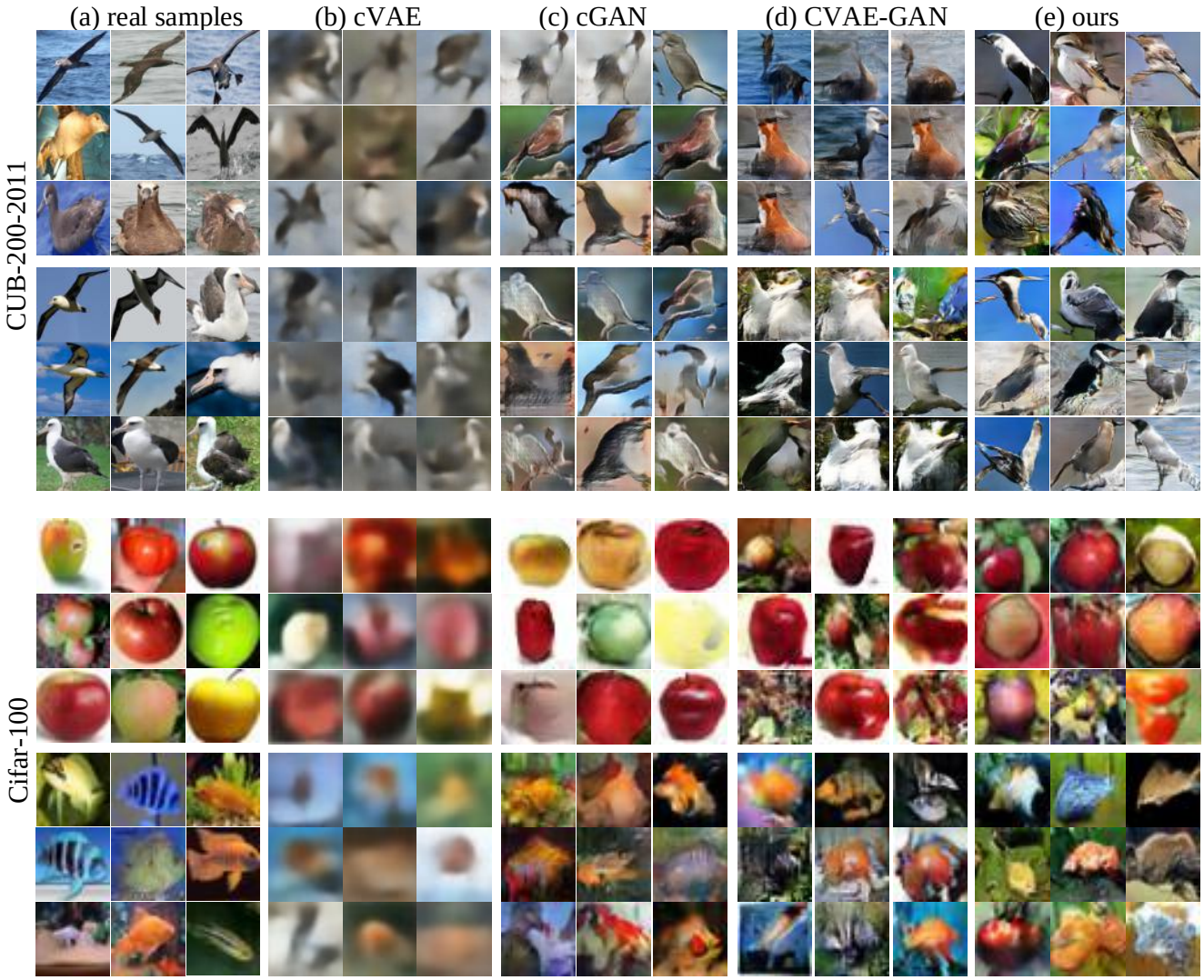


Figure 9. Visualization of generated images of different models on CUB-200-2011 and Cifar-100.