Appendix

A Derivations

A.1 InfoGAN optimizes Forward-KL Divergence

This section provides the derivation of Eq. (5) in the main paper. Consider $G(s, c)$ as a mapping function $G : \mathcal{S} \times \mathcal{C} \to \mathcal{X}$, where s and c denote nuisance and disentangled variables, respectively. Also, assuming G is a deterministic function of (s, c), the conditional distribution $p_G(x|s, c)$ can be approximated to a dirac distribution $\delta(x - G(s, c)) = \mathbb{1}\{x = G(s, c)\}$. Then, the marginal distribution $p_G(x)$ can be described as below:

$$
\begin{aligned}
p_G(x) &= \int_s \int_c p(s)p(c)p_G(x|s, c)dcds && \text{(A.1)}\\
&= E_{s\sim p(s),c\sim p(c)}[p_G(x|s, c)] && \text{(A.2)}\\
&= E_{s\sim p(s),c\sim p(c)}[\mathbb{1}\{x = G(s, c)\}]. && \text{(A.3)}
\end{aligned}
$$

Then, the variational lower-bound of mutual information optimized in InfoGAN [10] can be rewritten as follows:

$$
\begin{aligned}
\mathcal{R}_{\text{Info}}(G, q) &= E_{c\sim p(c),x\sim G(s,c)}[\log q_\phi(c|x)] + H(c),\\
&= \int_s p(s) \int_c p(c) \int_x p_G(x|s, c) \log q_\phi(c|x)dxdcds + H(c), && \text{(A.4)}\\
&= \int_s p(s) \int_c p(c) \int_x \mathbb{1}\{x = G(s, c)\} \log q_\phi(c|x)dxdcds + H(c), && \text{(A.5)}\\
&= \int_s p(s) \int_c p(c) \log q_\phi(c|G(s, c))dcds + H(c), && \text{(A.6)}\\
&= \int_s p(s) \int_c p(c) \log q_\phi(c|G(s, c))dcds - \int_c p(c) \log p(c)dc, && \text{(A.7)}\\
&= \int_s p(s) \int_c p(c) \log q_\phi(c|G(s, c))dcds - \int_c p(c) \log p(c)dc \int_s p(s)ds, && \text{(A.8)}\\
&= \int_s p(s) \int_c p(c) \log \frac{q_\phi(c|G(s, c))}{p(c)}dcds, && \text{(A.9)}\\
&= -E_{s\sim p(s)}[D_{KL}(p(c)||q_\phi(c|G(s, c))], && \text{(A.10)}
\end{aligned}
$$

where the Eq. (A.10) corresponds to Eq. (5) of the main paper. Similiarly, we can rewrite the distillation regularization $\mathcal{R}_{\text{ID}}(G)$ (Eq. (4) in the main paper)

as follows:

$$\begin{aligned}
\mathcal{R}_{\mathrm{ID}}(G) &= E_{c\sim q_\phi(c),x\sim G(s,c)}[\log q_\phi(c|x)] + H_{q_\phi}(c), \\
&= \int_s p(s) \int_c q_\phi(c) \int_x \mathbb{1}\{x = G(s,c)\} \log q_\phi(c|x) dx dc ds + H_{q_\phi}(c), \qquad \text{(A.11)} \\
&= \int_s p(s) \int_c q_\phi(c) \log q_\phi(c|G(s,c)) dc ds - \int_s p(s) \int_c q_\phi(c) \log q_\phi(c) dc ds, \qquad \text{(A.12)} \\
&= -E_{s\sim p(s)}[D_{KL}(q_\phi(c)||q_\phi(c|G(s,c))], \qquad \text{(A.13)}
\end{aligned}$$

where Eq. (A.13) corresponds to Eq. (7) in the main paper. As discussed in the paper, both Eq. (A.10) and (A.13) correspond to the *forward* KLD; regularization in InfoGAN $\mathcal{R}_{\mathrm{Info}}(G, q)$ (Eq. (A.10)) is optimized with respect to both the encoder q and the generator G, which is problematic due to the zero-avoiding characteristics of forward KLD and the potential mismatch between the true and generated data distributions. On the other hand, our method can effectively avoid this problem by optimizing Eq. (A.13) with only respect to the generator while encoder training is guided by *reverse* KLD using the true data distribution (Eq. (6) in the main paper).

A.2 cGAN implicitly maximizes $\mathcal{R}_{\mathrm{ID}}(G)$

In Section 5.3 of the main paper, we define cGAN as the baseline that also optimizes $\mathcal{R}_{\mathrm{ID}}(G)$ implicitly in its objective function. This section provides its detailed derivation. Formally, we consider cGAN that minimizes a Jensen-Shannon divergence (JSD) between two joint distributions $\mathrm{D}_{\mathrm{JS}}(p_d(x,c)||p_G(x,c))$, where $p_d(x,c) = p(x)q_\phi(c|x)$ and $p_G(x,c) = q_\phi(c)p_G(x|c) = q_\phi(c)\int_s p(s)p_G(x|s,c)ds$ denote real and fake joint distributions, respectively. Then, $\mathcal{R}_{\mathrm{ID}}(G)$ from $\mathrm{D}_{\mathrm{JS}}(p_d(x,c)||p_g(x,c))$ is derived as follows:

$$\begin{aligned}
&2\mathrm{D}_{\mathrm{JS}}(p_d(x,c)||p_g(x,c)) \qquad \text{(A.14)} \\
&= \mathrm{D}_{\mathrm{KL}}(p_d(x,c)||p_G(x,c)) + \mathrm{D}_{\mathrm{KL}}(p_G(x,c)||p_d(x,c)) \qquad \text{(A.15)} \\
&= \mathrm{D}_{\mathrm{KL}}(p_d(x,c)||p_G(x,c)) + \int_{c,x} p_G(x,c) \log \frac{p_G(x,c)}{p_d(x,c)} dx dc \qquad \text{(A.16)} \\
&= \mathrm{D}_{\mathrm{KL}}(p_d(x,c)||p_G(x,c)) + \int_{c,x} q_\phi(c)p_G(x|c) \log \frac{q_\phi(c)p_G(x|c)}{p(x)q_\phi(c|x)} dx dc \qquad \text{(A.17)} \\
&= \mathrm{D}_{\mathrm{KL}}(p_d(x,c)||p_G(x,c)) + \int_{c,x} q_\phi(c)p_G(x|c) \log \frac{p_G(x|c)}{p(x)} dx dc \\
&\quad + \int_{c,x} q_\phi(c)p_G(x|c) \log \frac{q_\phi(c)}{q_\phi(c|x)} dx dc \qquad \text{(A.18)} \\
&= \mathrm{D}_{\mathrm{KL}}(p_d(x,c)||p_G(x,c)) + \int_c q_\phi(c) \int_x p_G(x|c) \log \frac{p_G(x|c)}{p(x)} dx dc
\end{aligned}$$

$$+ \int_s p(s) \int_c q_\phi(c) \int_x p_G(x|s,c) \log \frac{q_\phi(c)}{q_\phi(c|x)} dx dc ds \tag{A.19}$$

$$= \mathrm{D_{KL}}(p_d(x,c)||p_G(x,c)) + \mathbb{E}_{c\sim q_\phi(c)}[\mathrm{D_{KL}}(p_G(x|c)||p(x)] - \mathcal{R}_{\mathrm{ID}}(G). \tag{A.20}$$

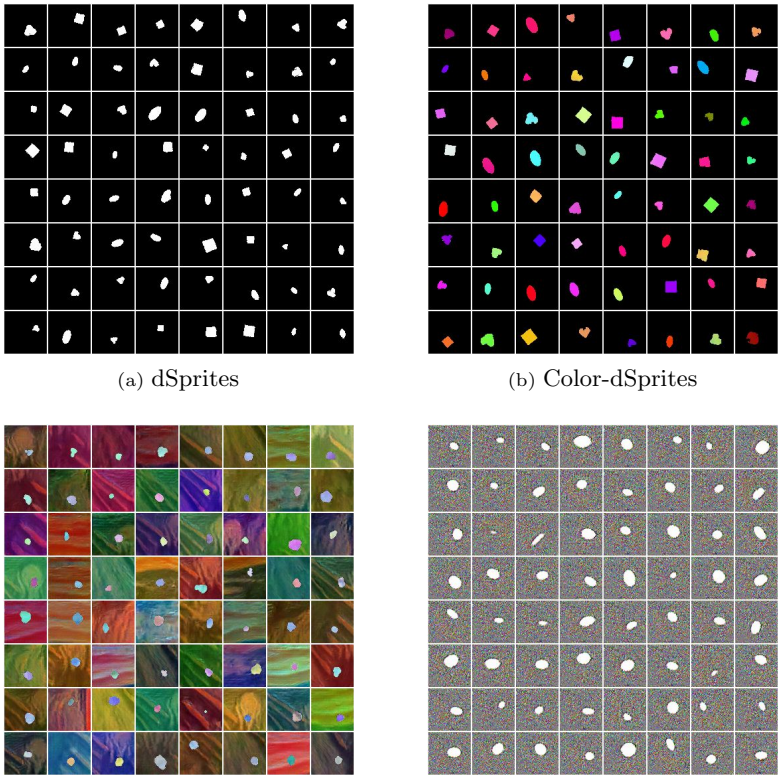
Eq. (A.20) implies that the cGAN objective also implicitly maximizes $\mathcal{R}_{\mathrm{ID}}(G)$. However, Eq. (A.20) is guaranteed only when the discriminator converges to (near-)optimal with respect to real and fake joint distributions, which makes the optimization of $\mathcal{R}_{\mathrm{ID}}(G)$ highly dependent on the quality of the discriminator. On the other hand, ID-GAN maximizes $\mathcal{R}_{\mathrm{ID}}(G)$ explicitly by directly computing $\mathcal{R}_{\mathrm{ID}}(G)$ from the learned encoder q_ϕ, which leads to a higher degree of alignment between the input latent code and the generated output (Table 3 and Figure 5 in the main paper).

B Additional Experiment Results

B.1 Additional Results on Synthetic Dataset

We present additional qualitative results on the synthetic dataset, which corresponds to Section 5.2 in the main paper. Figure A.1 presents the randomly generated images by the proposed ID-GAN. We observe that the generated images are sharp and realistic, capturing complex patterns in the background (Screem-dSprites and Noise-dSprites datasets). We also observe that it generates convincing foreground patterns, such as color and shape of the objects, while covering diverse and comprehensive patterns in real objects.

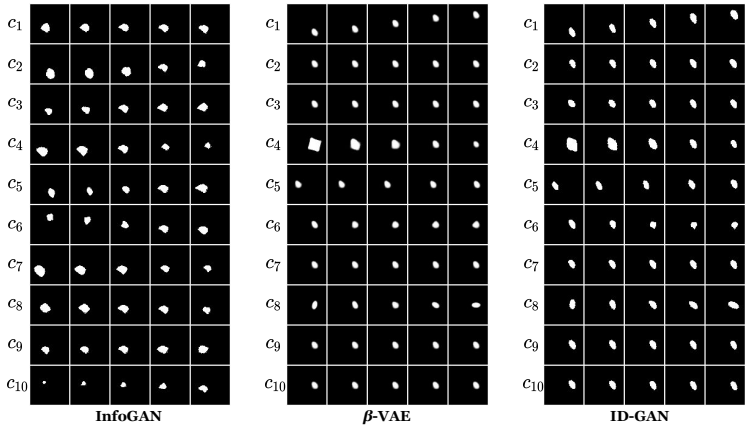
Figure A.2 and A.3 present additional qualitative comparison results with β-VAE and InfoGAN by manipulating the disentangled factors, which correspond to Figure 4 in the main paper. We observe that β-VAE captures the meaningful disentangled factors, such as location and color of the object, but overlooks several complex but important patterns in the background (Screem-dSprites and Noise-dSprites datasets) as well as foreground (*e.g.,* detailed shape and orientation). On the other hand, InfoGAN generates more convincing images by employing a more expressive decoder, but learns much more entangled representations (*e.g.,* changing location and color of the objects in Color-dSprites dataset). By combining the benefits of both approaches, ID-GAN successfully learns meaningful disentangled factors and generates realistic patterns.



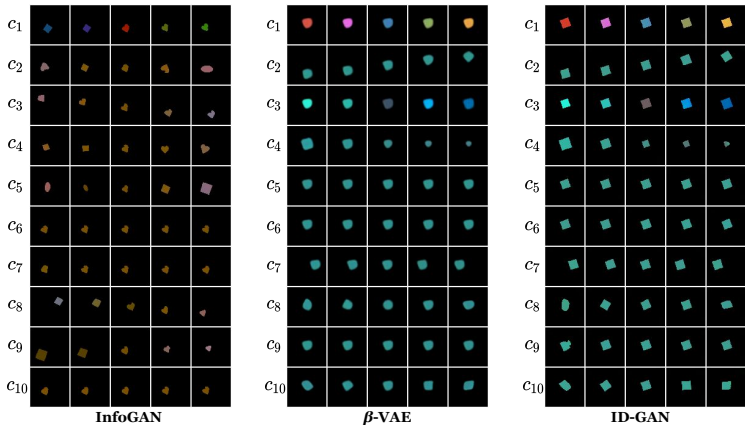
(a) dSprites (b) Color-dSprites

(c) Scream-dSprites (d) Noisy-dSprites

Fig. A.1. Random generated samples of ID-GAN on dSprites and its variants.

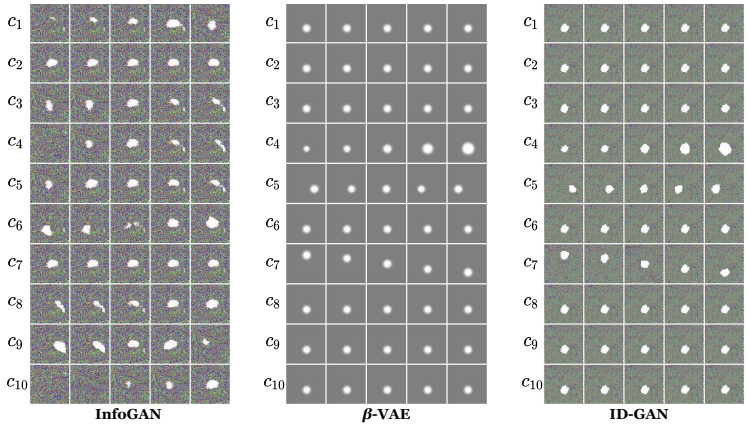


(a) dSprites

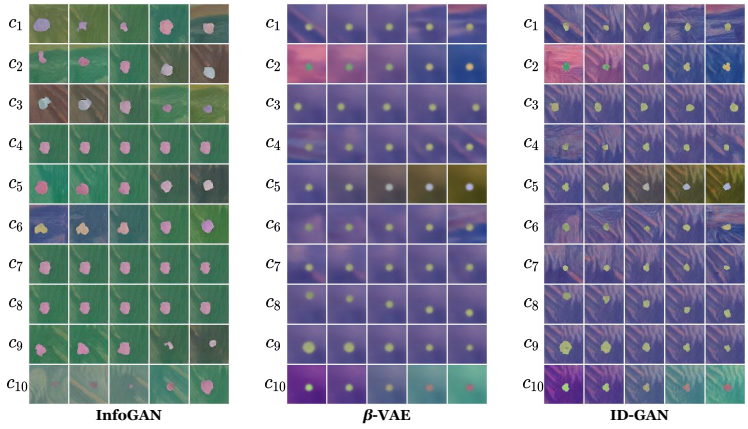


(b) Color-dSprites

Fig. A.2. Latent traversals of InfoGAN, β-VAE, and ID-GAN on (a) dSprites and (b) Color-dSprites datasets. Each c_j $(j = 1, \ldots, 10)$ represents a single dimension of c.



(a) Noisy-dSprites



(b) Scream-dSprites

Fig. A.3. Latent traversals of InfoGAN, β-VAE, and ID-GAN on (a) Noisy-dSprites and (b) Scream-dSprites datasets. Each c_j $(j = 1, \ldots, 10)$ represents a single dimension of c.

B.2 Additional Results on a Complex Dataset

Qualitative results of Table 4. Here we compare the qualitative samples generated by each model in Table 4, *i.e.,*. VAE, β-VAE, FactorVAE, GAN, InfoGAN, and ID-GAN. The qualitative results are shown in Figure A.4. Although VAE-based methods learn to represent global structures or salient factors of data in all datasets, generated samples are often blurry and lack textural or local details. On the other hand, GAN-based approaches (*i.e.,* GAN, InfoGAN and ID-GAN) generate sharp and realistic samples thanks to the implicit density estimation and expressive generators. However, as shown in Figure A.5, InfoGAN generally fails to capture meaningful disentangled factors into c since it exploits the nuisance variable s to encode the most salient factors of variations. On the other hand, ID-GAN successfully captures major disentangled factors into c while encoding only local details into the nuisance variable s.

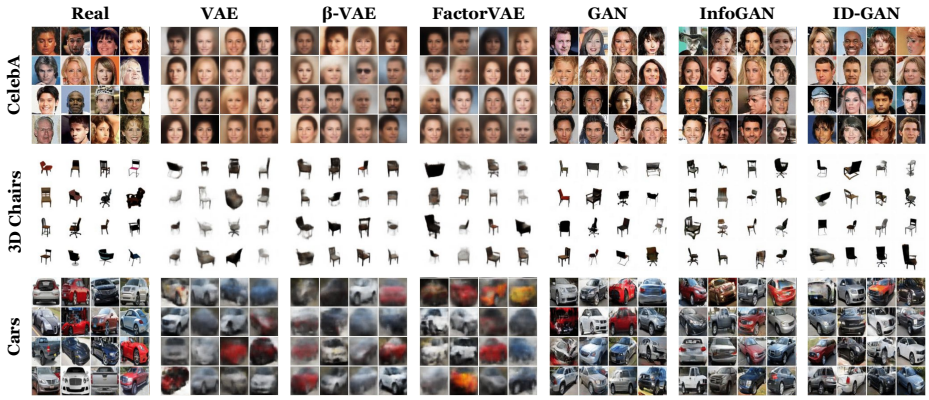


Fig. A.4. Random samples generated by VAE [46], β-VAE [18], FactorVAE [26], GAN [42], InfoGAN [10], and ID-GAN on CelebA, 3D Chairs, and Cars datasets (64×64).

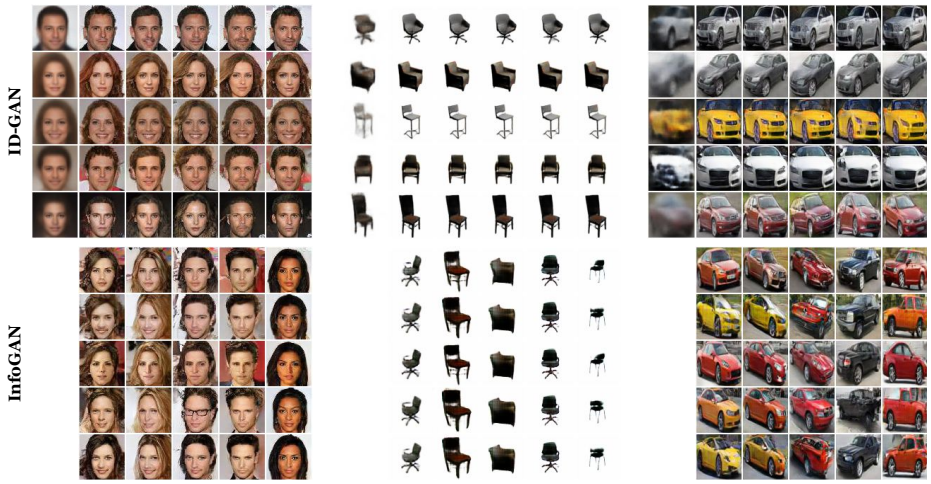


Fig. A.5. Analysis on the learned factors of variations in ID-GAN and InfoGAN. We sample 5 points for each c and s and visualize how they are used during the generative process of ID-GAN and InfoGAN. The samples in each row are generated from a single c with 5 different s. Also, we show the generated samples of β-VAE from each c at the first column of the panes of ID-GAN. As shown above, ID-GAN successfully learns to decode the global structures of data from c as β-VAE, while utilizing s as additional sources of variations for modeling local details. In InfoGAN, however, the most salient factors of variations are dominated by s while c acts as the nuisance.

Additional results on high-resolution synthesis. Here we provide more qualitative results of ID-GAN on high-resolution image synthesis (CelebA 256×256 and CelebA-HQ datasets). We first present the results on the CelebA-HQ dataset composed of mega-pixel images (1024×1024 pixels). Figure A.6 presents the randomly generated samples by ID-GAN. We observe that ID-GAN produces sharp and plausible samples on high resolution images, showing on par generation performance with the state-of-the-art GAN baseline [47] employed as a backbone network of ID-GAN. We argue that this is due to the separate decoder and generator scheme adopted in ID-GAN, which is hardly achievable in the VAE-based approaches using a factorized Gaussian decoder for explicit maximization of data log-likelihood.

Next, we analyze the learned factors of variations in ID-GAN by investigating the disentangled and nuisance variable c and s, respectively. Similarly to Figure 8 in the main paper, we compare the samples generated by fixing one latent variable while varying another. The results are summarized in Figure A.7. Similar to Figure 8, we observe that the disentangled variable c contributes to the most salient factors of variations (*e.g.,*azimuth, shape, or colour of face and hair, *etc..*) while the nuisance variable s contributes to the remaining fine details (*e.g.,*identity, hair style, expression, background, *etc..*). For instance, we observe that fixing the disentangled variable c leads to consistent global face structure (*e.g.,* black male facing slightly right (first column), blonde female facing slightly left (fourth column)), while fixing nuisance variable s leads to consistent details (*e.g.,* horizontally floating hair (third row), smiling expression (fourth and fifths rows)). These results suggest that the generator in ID-GAN is well-aligned with the VAE decoder to render the same disentangled variable c into similar observations, but with more expressive and realistic details by exploiting the nuisance variable s.

Finally, to further visualize the learned disentangled factors in c, we present the latent traversal results in Figure A.8 as an extension to Figures 1 and 9 in the main paper. We also visualize the results on CelebA 256 × 256 images in Figure A.9, where we observe a similar behavior.

Fig. A.6. Random samples generated by ID-GAN on the CelebA-HQ dataset (1024×1024). ID-GAN is based on VGAN architecture [47] and is trained to render learned disentangled representation c of β-VAE trained on much smaller 64×64 image resolution.

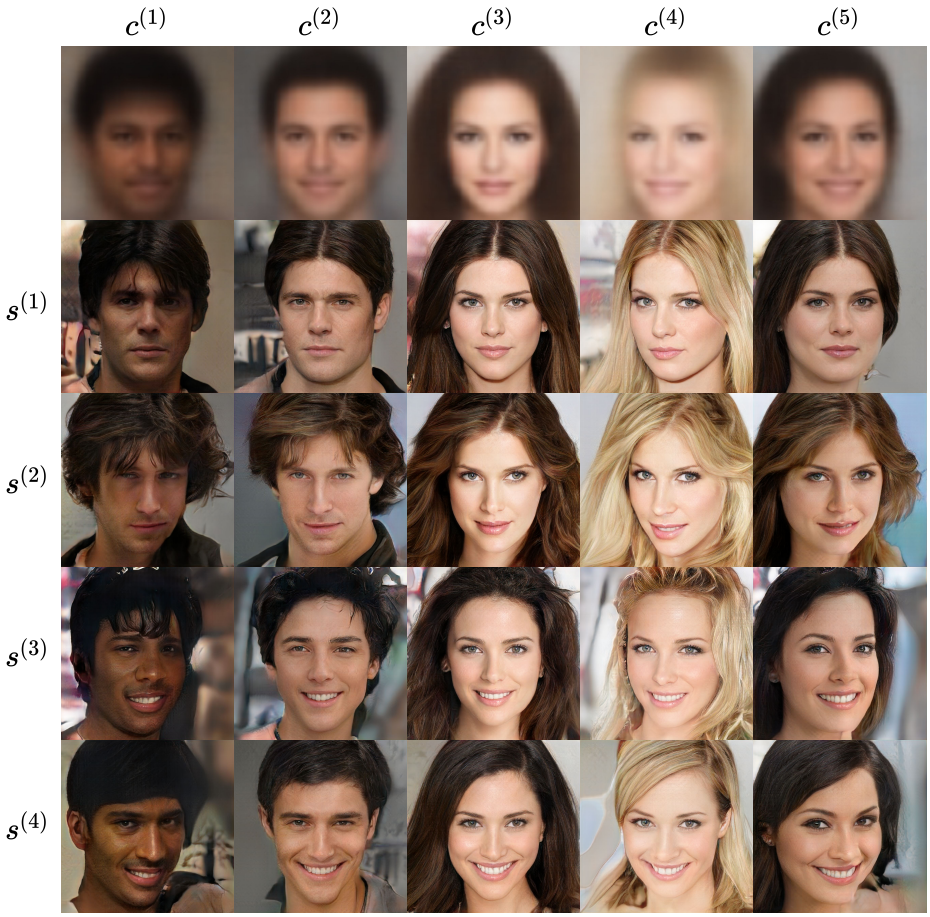


Fig. A.7. Similar visualizations as Figure 8, but on a more challenging CelebA-HQ (1024×1024) dataset. We can clearly observe the different contributions of disentangled variable c and nuisance variable s to the generative process $G(s, c)$; disentangled variable c captures the most salient factors of variations in the data (*e.g.,*azimuth and overall structure/color of face/hair are largely determined by c); nuisance variable s contributes to the remaining fine details (*e.g.,*identity, hair style, expression, background, *etc.*.).

Fig. A.8. Latent traversal results of ID-GAN on CelebA-HQ (1024 × 1024) dataset.

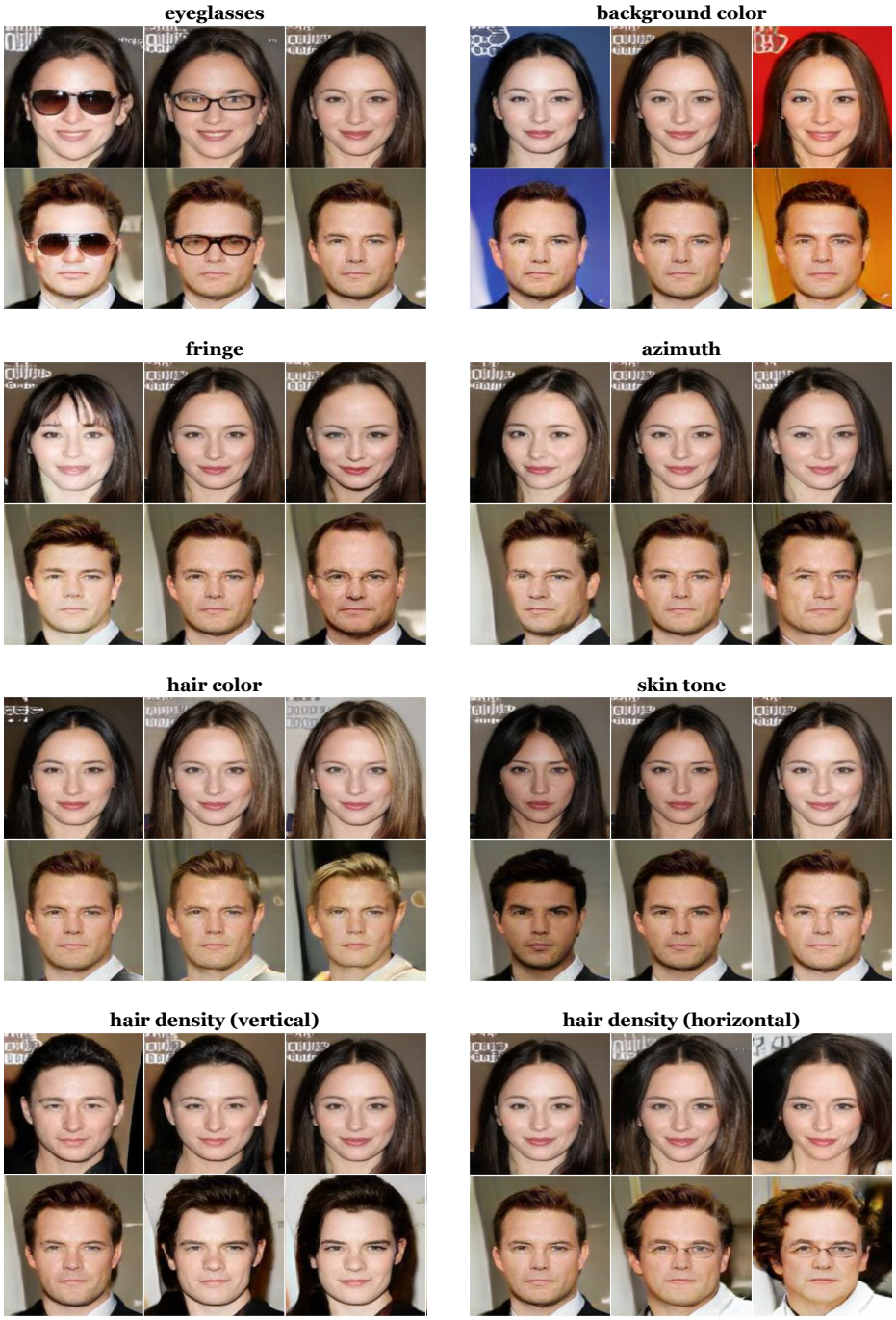


Fig. A.9. Latent traversals of ID-GAN on the CelebA dataset (256×256).

B.3 Sensitivity of Generation Performance (FID) on the Hyperparameter λ

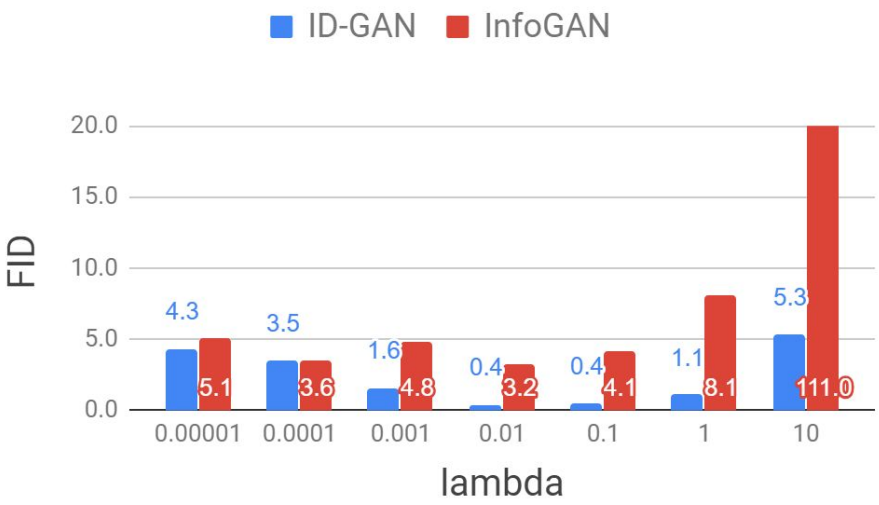


Fig. A.10. Sensitivity of the generation performance (FID) on λ.

To better understand the sensitivity of our model to its hyperparmeter (λ in Eq. (2)), we conduct an ablation study by measuring the generation performance (FID) of our models trained with various λ. Figure A.10 summarizes the results on the dSprite dataset. First, we observe that the proposed ID-GAN performs well over a wide range of hyper-parameters ($\lambda \in [0.001, 1]$) while the performance of InfoGAN is affected much sensitively to the choice of λ. Interestingly, increasing the λ in our method also leads to the improved generation quality over a certain range of λ. We suspect that it is because the information maximization in Eq. (4) using the pre-trained encoder also behaves as the perceptual loss [13,23,30], regularizing the generator to match the true distribution in more meaningful feature space (*i.e.,* disentangled representation).

C Implementation Details

C.1 Evaluation Metrics

FactorVAE Metric (FVM). FVM [26] measures the accuracy of a majority-vote classifier, where the encoder network to be evaluated is used for constructing the training data of this classifier. A single training data, or *vote*, is generated as follows: we first extract encoder outputs from the entire samples of a synthetic dataset; estimate empirical variances of each latent dimension from the extracted outputs; sort out collapsed latent dimensions of variances smaller than 0.05; synthesize 100 samples with a single factor fixed and the other factors varying randomly; extract encoder outputs from synthesized samples; compute variances of each latent dimension divided by the empirical variances computed beforehand; then finally get a single vote which is a pair of the index of the fixed factor and the index of the latent dimension with the smallest normalized variance. We generate 800 votes to train the majority-vote classifier and report its train accuracy as the metric score.

Mutual Information Gap (MIG). MIG [9] is an information-theoretic approach to measure the disentanglement of representations. Specifically, assuming K generative factors v_k $(k = 1, \dots, K)$ and D-dimensional latents z_j $(j = 1, \dots, D)$, it computes a normalized empirical mutual information $I(z_j; v_k)/H(v_k)$ to measure the information-theoretic preferences of z_j towards each v_k, or vice versa. Then, it aggregates the differences, or *gap*, between the top two preferences for each v_k and averages them to compute MIG, *i.e.* $\frac{1}{K}\sum_{k=1}^{K}\frac{1}{H(v_k)}(I(z_{j(k)}; v_k) - \min_{j \neq j(k)} I(z_j; v_k))$, where $I(z_{j(k)}; v_k) = \arg\max_j I(z_j; v_k)$. For implementation details, we directly follow the settings[1] in [37].

Fréchet Inception Distance (FID). We employ Fréchet Inception Distance (FID) to evaluate the generation quality of each model considered in our experiments. FID measures the fréchet distance [14] between two Gaussians, constructed by generated and real images, respectively, in the feature space of a pre-trained deep neural network. For each model, we compare 50,000 generated images and 50,000 real images to compute FID. For dSprites and its variants, we use a manually trained ConvNet trained to predict true generative factors of dSprites and its varaints. For the CelebA, 3D Chairs RGB, and Cars datasets, we use Inception V3 [49] pre-trained on the ImageNet [12] dataset. We use the publicly available code[2] to compute FID.

[1] https://github.com/google-research/disentanglement_lib
[2] https://github.com/mseitzer/pytorch-fid

C.2 Dataset

Table A.1. Descriptions on datasets.

Name	Description
dSprites [41]	737,280 binary 64x64 images of 2D sprites with 5 ground-truth factors, including shape (3), scale (6), orientation (40), x-position (32), and y-position (32).
Color-dSprites [7,37]	The sprite is filled with a random color. We randomly sample intensities of each color channel from 8 discrete values, linearly spaced between [0, 1].
Noisy-dSprites [37]	The background in each dSprites sample is filled with random uniform noise.
Scream-dSprites [37]	The background of each dSprites sample is replaced with a randomly-cropped patch of *The Scream* painting and the sprite is colored with the inverted color of the patch over the pixel regions of the sprite.
CelebA [35], CelebA-HQ [24]	CelebA dataset contains 202,599 RGB images of celebrity faces, which is composed of 10,177 identities, 5 landmark locations, and 40 annotated attributes of human faces. We use the *aligned&cropped* version of the dataset with the image size of 64×64 and 256×256. CelebA-HQ is the subset of the *in-the-wild* version of the CelebA dataset, which is composed of 30,000 RGB 1024×1024 high-resolution images.
3D Chairs [3]	86,366 RGB 64×64 images of chair CAD models with 1,393 types, 31 azimuths, and 2 elevations.
Cars [27]	16,185 RGB images of 196 classes of cars. We crop and resize each image into the size of 64×64 using the bounding-box annotations provided.

C.3 Architecture

Table A.2. Architectures of β-VAE and FactorVAE for all datasets. Note that Discriminator is needed only when training FactorVAE.

Encoder	**Decoder**	**Discriminator**
Input: 64 × 64 × # channels	Input: $\mathbb{R}^{10}$	FC 1000, leaky ReLU
4×4 conv 32, ReLU, stride 2	FC 256, ReLU	FC 1000, leaky ReLU
4×4 conv 32, ReLU, stride 2	FC 4×4×64, ReLU	FC 1000, leaky ReLU
4×4 conv 64, ReLU, stride 2	4×4 upconv 64, ReLU, stride 2	FC 1000, leaky ReLU
4×4 conv 64, ReLU, stride 2	4×4 upconv 32, ReLU, stride 2	FC 1000, leaky ReLU
FC 256, FC 2×10	4×4 upconv 32, ReLU, stride 2	FC 1000, leaky ReLU
	4×4 upconv # channels, stride 2	FC 2

Table A.3. Architectures of Generator and Discriminator networks for ID-GAN and InfoGAN on dSprites, Color-dSprites, Noisy-dSprites, and Scream-dSprites datasets. The encoder and the decoder networks are specified in Table A.2.

Generator	**Discriminator**
Input: $\mathbb{R}^{10}$	Input: 64 × 64 × # channels
FC 256, ReLU	4×4 conv 32, ReLU, stride 2
FC 4×4×64, ReLU	4×4 conv 32, ReLU, stride 2
4×4 upconv 64, ReLU, stride 2	4×4 conv 64, ReLU, stride 2
4×4 upconv 32, ReLU, stride 2	4×4 conv 64, ReLU, stride 2
4×4 upconv 32, ReLU, stride 2	FC 256, FC 1
4×4 upconv # channels, stride 2	

Table A.4. Architectures of Generator and Discriminator networks for ID-GAN and InfoGAN on CelebA, 3D Chairs, and Cars (64 × 64) datasets. We directly follow the architecture proposed in [42]. The encoder and the decoder networks are specified in Table A.2.

Generator	**Discriminator**
Input: $\mathbb{R}^{20+256}$	Input: 64 × 64 × 3
FC 4×4×512	3×3 conv 64, stride 1
ResBlock 512, NN Upsampling	ResBlock 64, AVG Pooling
ResBlock 256, NN Upsampling	ResBlock 128, AVG Pooling
ResBlock 128, NN Upsampling	ResBlock 256, AVG Pooling
ResBlock 64, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 64, 4×4 conv 3, stride 1	FC 1

Table A.5. Architectures of Generator and Discriminator networks for ID-GAN (w/o distill), cGAN, and ID-GAN on the CelebA (128×128) dataset. We directly follow the architecture proposed in [42]. The encoder and the decoder networks are specified in Table A.2.

Generator	**Discriminator**
Input: $\mathbb{R}^{20+256}$	Input: 128 × 128 × 3
FC 4×4×512	3×3 conv 64, stride 1
ResBlock 512, NN Upsampling	ResBlock 64, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 128, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 256, AVG Pooling
ResBlock 256, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 128, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 128, 4×4 conv 3, stride 1	FC 1

Table A.6. Architectures of Generator and Discriminator networks for ID-GAN on the CelebA (256×256) dataset. We directly follow the architecture proposed in [42]. The encoder and the decoder networks are specified in Table A.2.

Generator	**Discriminator**
Input: $\mathbb{R}^{20+256}$	Input: 256 × 256 × 3
FC 4×4×512	3×3 conv 64, stride 1
ResBlock 512, NN Upsampling	ResBlock 64, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 128, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 256, AVG Pooling
ResBlock 256, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 128, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 64, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 64, 4×4 conv 3, stride 1	FC 1

Table A.7. Architectures of Generator and Discriminator networks for ID-GAN on the CelebA-HQ (1024×1024) dataset. We directly follow the architecture proposed in [47]. The encoder and the decoder networks are specified in Table A.2.

Generator	**Discriminator**
Input: $\mathbb{R}^{20+256}$	Input: 1024 × 1024 × 3
FC 4×4×512	ResBlock 16, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 32, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 64, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 128, AVG Pooling
ResBlock 512, NN Upsampling	ResBlock 256, AVG Pooling
ResBlock 256, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 128, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 64, NN Upsampling	ResBlock 512, AVG Pooling
ResBlock 32, NN Upsampling	1×1 conv 2×512, Sampling 512
ResBlock 16,4×4 conv 3, stride 1	FC 1