Supplementary Material for Generative Adversarial Image Synthesis with Decision Tree Latent Controller

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

In this supplementary material, we provide additional analysis in Section A, give details on the experimental setup in Section B, and provide extended results in Section C. We provide other supplementary materials including demo videos at http://www.kecl.ntt.co.jp/people/kaneko.takuhiro/projects/dtlc-gan/.

A. Additional Analysis

A.1. Representation Comparison on Simulated Data

To clarify the limitation of the InfoGANs compared in Figure 5, we conducted experiments on simulated data. In particular, we used simulated data that are hierarchically sampled in the 2D space and have globally ten categories and locally two categories. When sampling data, we first randomly selected a global position from ten candidates that are equally spaced around a circle of radius 2. We then randomly selected a local position from two candidates that are rotated by 0.05 radians in clockwise and anticlockwise directions from the global position. Based on this local position, we sampled data from a Gaussian distribution of a standard deviation of 0.1.

We compared models that are similar to those compared in Figure 5. As the proposed model, we used the **DTLC^2-GAN**, where $k_1 = 10$ and $k_2 = 2$. In this model, $\hat{\boldsymbol{c}}_2$, the dimension of which is $10 \times 2 = 20$, is given to the G. For comparison, we used two models in which latent code dimensions are also 20 but not hierarchical. One is the **InfoGAN**$_{1\times20}$, which has one code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 20, p = 0.05)$, and the other is the **InfoGAN**$_{2\times10}$, which has two codes $\boldsymbol{c}_1^1, \boldsymbol{c}_1^2 \sim \mathrm{Cat}(K = 10, p = 0.1)$. For **DTLC^2-GAN**, we also compared the **DTLC^2-GANs** with and without curriculum learning.

We show the results in Figure 10. The results indicate that **InfoGANs** (b, c) and **DTLC^2-GAN** without curriculum learning (d) tend to cause unbalanced or non-hierarchical clustering. In contrast, the **DTLC^2-GAN** with curriculum learning (e) succeeds in capturing hierarchical structures, i.e., the first-layer codes captured global ten points, whereas the second-layer codes captured local two points for each global position.

A.2. Visual Interpretability Analysis

To clarity the benefit of learned representations, we conducted two XAB tests. For each test, we compared the fourth-layer models (DTLC^4-GANs or DTLC^4-WGAN-GPs) with and without curriculum learning.

- **Test I: Difference Interpretability Analysis**
  To confirm whether $\hat{\boldsymbol{c}}_L$ is more interpretable than $\boldsymbol{z}$, we compared the generated images (X) with the images generated from latent variables in which one dimension of $\boldsymbol{z}$ is changed (A) and one dimension of $\hat{\boldsymbol{c}}_4$ is changed (B). The changed dimension of $\boldsymbol{z}$ or $\hat{\boldsymbol{c}}_4$ was randomly chosen. We asked participants which difference is more interpretable or even.

- **Test II: Semantic Similarity Analysis**
  To confirm whether $\hat{\boldsymbol{c}}_L$ is hierarchically interpretable, we compared the generated images (X) with the images generated from latent variables in which one dimension of $\boldsymbol{c}_2$ is varied (A) and one dimension of $\boldsymbol{c}_4$ is varied (B). For each case, we fixed the higher layer codes. The changed dimension of $\boldsymbol{c}_2$ or $\boldsymbol{c}_4$ was randomly chosen. The lower layer codes were also randomly chosen. We asked participants which is semantically similar or even.

To eliminate bias in individual samples, we showed 25 samples at the same time. To eliminate bias in the order of stimuli, the order (AB or BA) was randomly selected. We show the user interfaces in Figure 11.

We summarize the results in Tables 5. In (a) and (b), we list the results of tests I and II, respectively, using the **DTLC^4-GAN**$_{\mathrm{WS}}$, which were used for the experiments discussed in Figure 7. The results of test I indicate that $\hat{\boldsymbol{c}}_L$ is more interpretable than $\boldsymbol{z}$ regardless of curriculum learning. We argue that this is because $\boldsymbol{z}$ does not have any constraints on a structure and may be used by the G in a highly entangled manner. The results of test II indicate that

Model	$\boldsymbol{z}$	even	$\hat{\boldsymbol{c}}_4$
W/o curriculum	0.0	1.0 ± 1.0	$\mathbf{99.0 \pm 1.0}$
W/ curriculum	0.0	1.0 ± 1.0	$\mathbf{99.0 \pm 1.0}$

*Number of collected answers is 400

(a) Test I for DTLC^4-GAN$_{\mathrm{WS}}$ on CIFAR-10

Model	$\boldsymbol{c}_2$	even	$\boldsymbol{c}_4$
W/o curriculum	22.4 ± 3.9	41.3 ± 4.6	36.2 ± 4.5
W/ curriculum	3.6 ± 1.7	17.8 ± 3.5	$\mathbf{78.7 \pm 3.8}$

*Number of collected answers is 450

(b) Test II for DTLC^4-GAN$_{\mathrm{WS}}$ on CIFAR-10

Model	$\boldsymbol{c}_2$	even	$\boldsymbol{c}_4$
W/o curriculum	18.0 ± 4.4	31.3 ± 5.3	50.7 ± 5.7
W/ curriculum	4.7 ± 2.4	12.0 ± 3.7	$\mathbf{83.3 \pm 4.2}$

*Number of collected answers is 300

(c) Test II for DTLC^4-WGAN-GP on CIFAR-10

Model	$\boldsymbol{c}_2$	even	$\boldsymbol{c}_4$
W/o curriculum	21.7 ± 4.7	38.3 ± 5.5	40.0 ± 5.6
W/ curriculum	17.0 ± 4.3	24.0 ± 4.9	$\mathbf{59.0 \pm 5.6}$

*Number of collected answers is 300

(d) Test II for DTLC^4-WGAN-GP$_{\mathrm{WS}}$ on CIFAR-10

Model	$\boldsymbol{c}_2$	even	$\boldsymbol{c}_4$
W/o curriculum	13.2 ± 4.2	53.6 ± 6.2	33.2 ± 5.9
W/ curriculum	2.4 ± 1.9	17.2 ± 4.7	$\mathbf{80.4 \pm 5.0}$

*Number of collected answers is 250

(e) Test II on DTLC^4-WGAN-GP on Tiny ImageNet

Table 5. Average preference score (%) with 95% confidence intervals. We compared fourth-layer models (DTLC^4-GANs or DTLC^4-WGAN-GPs) with and without curriculum learning.

representations learned with curriculum learning are hierarchically categorized in a better way in terms of semantics than those without it. The results support the effectiveness of the proposed curriculum learning method.

We also conducted test II (semantic similarity analysis) for all the **DTLC^4-WGAN-GPs** discussed in Section 6.3. We summarize the results in Table 5(c)–(e). We observed a similar tendency as those of **DTLC^4-GAN$_{\mathrm{WS}}$**.

A.3. Unsupervised Learning on Complex Dataset

Although, in Section 6.1, we mainly analyzed unsupervised settings on the MNIST dataset, which is relatively simple, we can learn hierarchical representations in an unsupervised manner even in more complex datasets. However, in this case, learning targets depend on the initialization because such datasets can be categorized in various ways. We illustrate this in Figure 12. We also evaluated the DTLC-WGAN-GP in unsupervised settings on the CIFAR-10 and Tiny ImageNet datasets. See Section 6.3 for details.

B. Details on Experimental Setup

In this section, we describe the network architectures and training scheme for each dataset. We designed the network architectures and training scheme on the basis of techniques introduced for the InfoGAN [1]. The D and $Q_1, \cdots, Q_L$ share all convolutional layers (Conv.), and one fully connected layer (FC.) is added to the final layer for $Q_1, \cdots, Q_L$. This means that the difference in the calculation cost for the GAN and DTLC-GAN is negligibly small. For discrete code $\hat{\boldsymbol{c}}_l^m$, we represented Q_l^m as softmax nonlinearity. For continuous code $\hat{\boldsymbol{c}}_l^m$, we parameterized Q_l^m through a factored Gaussian.

In most of the experiments we conducted, we designed the network architectures and training scheme on the basis of the techniques introduced for the DCGAN [9] and did not use the state-of-the-art GAN training techniques to evaluate whether the DTLC-GAN works well without relying on such techniques. To downscale and upscale, we respectively used convolutions (Conv. $\downarrow$) and backward convolutions (Conv. $\uparrow$), i.e., fractionally strided convolutions, with stride 2. As activation functions, we used rectified linear units (ReLUs) [8] for the G, while we used leaky rectified linear units (LReLUs) [7, 10] for the D. We applied batch normalization (BNorm) [5] to all the layers except the generator output layer and discriminator input layer. We trained the networks using the Adam optimizer [6] with a mini-batch of size 64. The learning rate was set to 0.0002 for the $D/Q_1, \cdots, Q_L$ and to 0.001 for the G. The momentum term β_1 was set to 0.5.

To demonstrate that our contributions are orthogonal to the state-of-the-art GAN training techniques, we also tested the DTLC-WGAN-GP (our DTLC-GAN with the WGAN-GP ResNet [4]) discussed in Section 6.3. We used similar network architectures and training scheme as the WGAN-GP ResNet, except for the extended parts.

The details for each dataset are given below.

B.1. MNIST

The DTLC^L-GAN network architectures for the MNIST dataset, which were used for the experiments discussed in Section 6.1, are shown in Table 6. As a pre-process, we normalized the pixel value to the range $[0, 1]$. In the generator output layers, we used the Sigmoid function. We used the **DTLC^L-GAN**, where $k_1 = 10$ and $k_2, \cdots, k_L = 2$, i.e., which has one discrete code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 10, p = 0.1)$ in the first layer and N_l discrete codes $\boldsymbol{c}_l^n \sim \mathrm{Cat}(K = 2, p = 0.5)$ in the lth layer where $l = (2, \cdots, L)$, $n = (1, \cdots, N_L)$, and $N_l = \prod_{i=1}^{l-1} k_i$. We added $\hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$ to the generator input. The trade-off parameters $\lambda_1, \cdots, \lambda_L$ were set to 0.1. We trained the networks for 1×10^4 iterations in unsupervised settings. As a curriculum for $\hat{\boldsymbol{c}}_l$ ($l = 2, \cdots, L$), we added regularization $-\lambda_l \mathcal{L}_{\mathrm{HCMI}}(G, Q_l)$ and sampling after $2(l-1) \times 10^3$ iterations.

Generator G
Input $\boldsymbol{z} \in \mathbb{R}^{64} + \hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$
1024 FC., BNorm, ReLU
$7 \cdot 7 \cdot 128$ FC., BNorm, ReLU
4×4 64 Conv. $\uparrow$, BNorm, ReLU
4×4 1 Conv. $\uparrow$, Sigmoid

Discriminator D / **Auxiliary Function** $Q_1, \cdots, Q_L$
Input 28×28 1 gray image
4×4 64 Conv. $\downarrow$, LReLU
4×4 128 Conv. $\downarrow$, BNorm, LReLU
1024 FC., BNorm, LReLU
FC. output for D
[128 FC., BNorm, LReLU]-FC. output for $Q_1, \cdots, Q_L$

Table 6. DTLC^L-GAN network architectures used for MNIST

B.2. CIFAR-10

The DTLC^L-GAN network architectures for the CIFAR-10 dataset, which were used for the experiments discussed in Section 6.2, are shown in Table 7. As a pre-process, we normalized the pixel value to the range $[-1, 1]$. In the generator output layers, we used the Tanh function. We used the **DTLC^L-GAN$_{\mathrm{WS}}$**, where $k_1 = 10$ and $k_2, \cdots, k_L = 3$, i.e., which has one ten-dimensional discrete code $\boldsymbol{c}_1^1$ in the first layer and N_l discrete codes $\boldsymbol{c}_l^n \sim \mathrm{Cat}(K = 3, p = \frac{1}{3})$ in the lth layer where $l = (2, \cdots, L)$, $n = (1, \cdots, N_l)$, and $N_l = \prod_{i=1}^{l-1} k_i$. We added $\hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$ to the generator input. We used the supervision (i.e., class labels) for $\boldsymbol{c}_1^1$. The trade-off parameters $\lambda_1, \cdots, \lambda_L$ were set to 1. We trained the networks for 1×10^5 iterations in weakly supervised settings. As a curriculum for $\hat{\boldsymbol{c}}_l$ $(l = 3, \cdots, L)$, we added regularization $-\lambda_l \mathcal{L}_{\mathrm{HCMI}}(G, Q_l)$ and sampling after $2(l-2) \times 10^4$ iterations.

B.3. DTLC-WGAN-GP

The DTLC^L-WGAN-GP network architectures for the CIFAR-10 and Tiny ImageNet datasets, which were used for the experiments discussed in Section 6.3, are similar to the WGAN-GP ResNet used in a previous paper [4], except for the extended parts. We used the **DTLC^L-WGAN-GP**, where $k_1 = 10$ and $k_2, \cdots, k_L = 3$, i.e., which has one ten-dimensional discrete code $\boldsymbol{c}_1^1$ in the first layer and N_l discrete codes $\boldsymbol{c}_l^n \sim \mathrm{Cat}(K = 3, p = \frac{1}{3})$ in the lth layer where $l = (2, \cdots, L)$, $n = (1, \cdots, N_l)$, and $N_l = \prod_{i=1}^{l-1} k_i$. Following the AC-WGAN-GP ResNet implementation [4], we used conditional batch normalization (CBN) [2, 3] to make the G conditioned on the codes $\hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$. CBN has two parameters, i.e., gain parameter γ_j and bias parameter b_j, for each category, where $j = 1, \cdots, \prod_{l=1}^{L} k_l$. As curriculum for sampling, in learning the higher layer codes, we used γ_j and b_j averaged over those for the related lower layer node codes.

In unsupervised settings, we sampled $\boldsymbol{c}_1^1$ from categorical distribution $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 10, p = 0.1)$. The trade-off parameters $\lambda_1, \cdots, \lambda_L$ were set to 1. We trained the networks for 1×10^5 iterations. As a curriculum for $\hat{\boldsymbol{c}}_l$ $(l = 2, \cdots, L)$, we added regularization $-\lambda_l \mathcal{L}_{\mathrm{HCMI}}(G, Q_l)$ and sampling after $2(l-1) \times 10^4$ iterations.

Generator G
Input $\boldsymbol{z} \in \mathbb{R}^{128} + \hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$
$4 \cdot 4 \cdot 512$ FC., BNorm, ReLU
4×4 256 Conv. $\uparrow$, BNorm, ReLU
4×4 128 Conv. $\uparrow$, BNorm, ReLU
4×4 64 Conv. $\uparrow$, BNorm, ReLU
3×3 3 Conv., Tanh

Discriminator D / **Auxiliary Function** $Q_1, \cdots, Q_L$
Input 32×32 3 color image
3×3 64 Conv., LReLU, Dropout
4×4 128 Conv. $\downarrow$, BNorm, LReLU, Dropout
3×3 128 Conv., BNorm, LReLU, Dropout
4×4 256 Conv. $\downarrow$, BNorm, LReLU, Dropout
3×3 256 Conv., BNorm, LReLU, Dropout
4×4 512 Conv. $\downarrow$, BNorm, LReLU, Dropout
3×3 512 Conv., BNorm, LReLU, Dropout
FC. output for D
[128 FC., BNorm, LReLU, Dropout]- FC. output for $Q_1, \cdots, Q_L$

Table 7. DTLC^L-GAN network architectures used for CIFAR-10

In weakly supervised settings, we used the supervision (i.e., class labels) for $\boldsymbol{c}_1^1$. The $\lambda_1, \cdots, \lambda_L$ were set to 1. We trained the networks for 1×10^5 iterations. As a curriculum for $\hat{\boldsymbol{c}}_l$ $(l = 3, \cdots, L)$, we added regularization $-\lambda_l \mathcal{L}_{\mathrm{HCMI}}(G, Q_l)$ and sampling after $2(l-2) \times 10^4$ iterations.

B.4. 3D Faces

The DTLC^L-GAN network architectures for the 3D Faces dataset, which were used for the experiments discussed in Section 6.4, are shown in Table 8. As a pre-process, we normalized the pixel value to the range $[0, 1]$. In the generator output layers, we used the Sigmoid function. We used the **DTLC^2-GAN**, where $k_1 = 5$ and $k_2 = 1$, i.e., which has one discrete code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 5, p = 0.2)$ in the first layer and five continuous codes $\boldsymbol{c}_2^1, \cdots, \boldsymbol{c}_2^5 \sim \mathrm{Unif}(-1, 1)$ in the second layer. We added $\hat{\boldsymbol{c}}_2 \in \mathbb{R}^{\prod_{l=1}^{2} k_l}$ to the generator input. The trade-off parameters λ_1 and λ_2 were set to 1. We trained the networks for 1×10^4 iterations in unsupervised settings. As a curriculum for $\hat{\boldsymbol{c}}_2$, we added regularization $-\lambda_2 \mathcal{L}_{\mathrm{HCMI}}(G, Q_2)$ and sampling after 2×10^3 iterations.

B.5. CelebA

The DTLC^L-GAN network architectures for the CelebA dataset, which were used for the experiments discussed in Section 6.5, are shown in Table 9. As a pre-process, we

Generator G
Input $\boldsymbol{z} \in \mathbb{R}^{128} + \hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$
1024 FC., BNorm, ReLU
$8 \cdot 8 \cdot 128$ FC., BNorm, ReLU
4×4 64 Conv. $\uparrow$, BNorm, ReLU
4×4 1 Conv. $\uparrow$, Sigmoid

Discriminator D / **Auxiliary Function** $Q_1, \cdots, Q_L$
Input 32×32 1 gray image
4×4 64 Conv. $\downarrow$, LReLU
4×4 128 Conv. $\downarrow$, BNorm, LReLU
1024 FC., BNorm, LReLU
FC. output for D
[128 FC., BNorm, LReLU]-FC. output for $Q_1, \cdots, Q_L$

Table 8. DTLC^L-GAN network architectures used for 3D Faces

Generator G
Input $\boldsymbol{z} \in \mathbb{R}^{128} + \hat{\boldsymbol{c}}_L \in \mathbb{R}^{1+\prod_{l=2}^{L} k_l}$
$4 \cdot 4 \cdot 512$ FC., BNorm, ReLU
4×4 256 Conv. $\uparrow$, BNorm, ReLU
4×4 128 Conv. $\uparrow$, BNorm, ReLU
4×4 64 Conv. $\uparrow$, BNorm, ReLU
4×4 3 Conv. $\uparrow$, Tanh

Discriminator D / **Auxiliary Function** $Q_1, \cdots, Q_L$
Input 64×64 3 color image
4×4 64 Conv. $\downarrow$, LReLU
4×4 128 Conv. $\downarrow$, BNorm, LReLU
4×4 256 Conv. $\downarrow$, BNorm, LReLU
4×4 512 Conv. $\downarrow$, BNorm, LReLU
FC. output for D
[128 FC., BNorm, LReLU]-FC. output for $Q_1, \cdots, Q_L$

Table 9. DTLC^L-GAN network architectures used for CelebA

normalized the pixel value to the range $[-1, 1]$. In the generator output layers, we used the Tanh function. We used the **DTLC^L-GAN$_{\mathrm{WS}}$**, where $k_1 = 2$ and $k_2, \cdots, k_L = 3$, particularly where hierarchical representations are learned only for the attribute presence state. Therefore, $N_2 = 1$ and N_l $(l = 3, \cdots, L)$ is calculated as $N_l = \prod_{i=2}^{l-1} k_i$. This model has one two-dimensional discrete code in the first layer and N_l discrete codes $\boldsymbol{c}_l^n \sim \mathrm{Cat}(K = 3, p = \frac{1}{3})$ in the lth layer where $l = (2, \cdots, L)$ and $n = (1, \cdots, N_l)$. We added $\hat{\boldsymbol{c}}_L \in \mathbb{R}^{1+\prod_{l=2}^{L} k_l}$ to the generator input. We used the supervision (i.e., an attribute label) for $\boldsymbol{c}_1^1$. The trade-off parameters $\lambda_1, \cdots, \lambda_L$ were set to 1, 0.1, and 0.04 for bangs, glasses, and smiling, respectively. We trained the networks for 5×10^4 iterations in weakly supervised settings. As a curriculum for $\hat{\boldsymbol{c}}_l$ $(l = 3, \cdots, L)$, we added regularization $-\lambda_l \mathcal{L}_{\mathrm{HCMI}}(G, Q_l)$ and sampling after $2(l-2) \times 10^4$ iterations.

Generator G
Input $\boldsymbol{z} \in \mathbb{R}^{256} + \hat{\boldsymbol{c}}_L \in \mathbb{R}^{\prod_{l=1}^{L} k_l}$
128 FC., ReLU
128 FC., ReLU
2 FC.

Discriminator D / **Auxiliary Function** $Q_1, \cdots, Q_L$
Input 2D simulated data
(scaled by factor 4 (roughly scaled to range $[-1, 1]$))
128 FC. ReLU
128 FC. ReLU
FC. output for D
[128 FC., ReLU]-FC. output for $Q_1, \cdots, Q_L$

Table 10. DTLC^L-GAN network architectures used for simulated data

B.6. Simulated Data

The DTLC^L-GAN network architectures for the simulated data used for the experiments discussed in Section A.1, are shown in Table 10. As a pre-process, we scaled the discriminator input by factor 4 (roughly scaled to range $[-1, 1]$). We used the **DTLC^2-GAN**, where $k_1 = 10$ and $k_2 = 2$, i.e., which has one discrete code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 10, p = 0.1)$ in the first layer and ten discrete codes $\boldsymbol{c}_2^1, \cdots, \boldsymbol{c}_2^{10} \sim \mathrm{Cat}(K = 2, p = 0.5)$ in the second layer. We added $\hat{\boldsymbol{c}}_2 \in \mathbb{R}^{\prod_{l=1}^{2} k_l}$ to the generator input. The trade-off parameters λ_1 and λ_2 were set to 1. We trained the networks using the Adam optimizer with a mini-batch of size 512. The learning rate was set to 0.0001 for $D/Q_1, Q_2$ and G. The momentum term β_1 was set to 0.5. We trained the networks for 3×10^4 iterations in unsupervised settings. As a curriculum for $\hat{\boldsymbol{c}}_2$, we added regularization $-\lambda_2 \mathcal{L}_{\mathrm{HCMI}}(G, Q_2)$ and sampling after 2×10^4 iterations.

C. Extended Results

C.1. MNIST

We give extended results of Figure 6 in Figures 13–15. We used the **DTLC^4-GAN**, where $k_1 = 10$ and $k_2, k_3, k_4 = 2$. Figure 13 shows the generated image examples using the **DTLC^4-GAN** learned without a curriculum. Figure 14 shows the generated image examples using the **DTLC^4-GAN** learned only with the curriculum for regularization. Figure 15 shows the generated image examples using the **DTLC^4-GAN** learned with the full curriculum (curriculum for regularization and sampling: proposed curriculum learning method). The former two **DTLC^4-GANs** (without the full curriculum) exhibited confusion between inner-layer and intra-layer disentanglement, while the **DTLC^4-GAN** with the full curriculum succeeded in avoiding confusion. The inner-category divergence evaluation on the basis of the SSIM in Figure 6 also supports these observations.

C.2. CIFAR-10

We give extended results of Figure 7 in Figures 16–18. We used the **DTLC^4-GAN$_{\mathrm{WS}}$**, where $k_1 = 10$ and $k_2, k_3, k_4 = 3$. We used class labels as supervision. Figure 16 shows the generated image samples using the **DTLC^4-GAN$_{\mathrm{WS}}$** learned without a curriculum. Figure 17 shows the generated image samples using the **DTLC^4-GAN$_{\mathrm{WS}}$** learned only with the curriculum for regularization. Figure 18 shows the generated image samples using the **DTLC^4-GAN$_{\mathrm{WS}}$** learned with the full curriculum (curriculum for regularization and sampling: proposed curriculum learning method). All models succeeded in learning disentangled representations in class labels since they are given as supervision; however, the former two **DTLC^4-GAN$_{\mathrm{WS}}$s** (without the full curriculum) exhibited confusion between inner-layer and intra-layer disentanglement from second- to fourth-layer codes. In contrast, the **DTLC^4-GAN$_{\mathrm{WS}}$** with the full curriculum succeeded in avoiding confusion. The inner-category divergence evaluation on the basis of the SSIM in Figure 7 also supports these observations.

We also show the results for the **DTLC^5-GAN$_{\mathrm{WS}}$**, where $k_1 = 10$ and $k_2, \cdots, k_5 = 3$, in Figure 19. In this model, a total of $10 \times 3 \times 3 \times 3 \times 3 = 810$ categories were learned in a weakly supervised manner.

C.3. DTLC-WGAN-GP

We show the generated image samples using the models discussed in Section 6.3, in Figure 20–22. We used the **DTLC^4-WGAN-GP**, where $k_1 = 10$ and $k_2, k_3, k_4 = 3$. In weakly supervised settings, we used class labels as supervision. Figure 20 shows the generated image samples using the **DTLC^4-WGAN-GP** on CIFAR-10 (unsupervised). Figure 21 shows the generated image samples using the **DTLC^4-WGAN-GP$_{\mathrm{WS}}$** on CIFAR-10 (weakly supervised). Figure 22 shows the generated image samples using the **DTLC^4-WGAN-GP** on Tiny ImageNet (unsupervised).

C.4. 3D Faces

We give extended results of Figure 8 in Figure 23. Similarly to Figure 8, we compared three models, the **InfoGAN$_{\mathrm{C5}}$**, which is the InfoGAN with five continuous codes $\boldsymbol{c}_1^1, \cdots, \boldsymbol{c}_1^5 \sim \mathrm{Unif}(-1, 1)$ (used in the InfoGAN study [1]), **InfoGAN$_{\mathrm{C1D1}}$**, which is the InfoGAN with one categorical code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 5, p = 0.2)$ and one continuous code $\boldsymbol{c}_1^2 \sim \mathrm{Unif}(-1, 1)$, and **DTLC^2-GAN**, which has one categorical code $\boldsymbol{c}_1^1 \sim \mathrm{Cat}(K = 5, p = 0.2)$ in the first layer and five continuous codes $\boldsymbol{c}_2^1, \cdots, \boldsymbol{c}_2^5 \sim \mathrm{Unif}(-1, 1)$ in the second layer. In the **InfoGAN$_{\mathrm{C5}}$** and **InfoGAN$_{\mathrm{C1D1}}$**, the individual codes tend to capture independent and exclusive semantic features because they have a flat relationship, while in the **DTLC^2-GAN**, lower layer codes learn category-specific (in this case, pose-specific) semantic features conditioned on higher layer codes.

C.5. CelebA

We show the generated image examples using the models discussed in Section 6.5, in Figures 24–26. We used the **DTLC^3-GAN$_{\mathrm{WS}}$**, where $k_1 = 2$ and $k_2, k_3 = 3$, and particularly hierarchical representations are learned only for the attribute presence state. We show the results for bangs, glasses, and smiling in Figures 24, 25, and 26, respectively. These results indicate that the DTLC-GAN$_{\mathrm{WS}}$ can learn attribute-specific hierarchical interpretable representations by only using the supervision of the binary indicator of attribute presence.

We also show the generated image examples and image retrieval examples using the **DTLC^4-GAN$_{\mathrm{WS}}$**, where $k_1 = 2$ and $k_2, k_3, k_4 = 3$, in Figures 27 and 28, respectively. We show the results for glasses. In this model, a total of $1 + 1 \times 3 \times 3 \times 3 = 28$ categories were learned in a weakly supervised setting. These results indicate that more detailed semantic features were captured in the lower layers. As quantitative evaluation of image retrieval, we calculated attribute-specific SSIM scores. The scores for $\boldsymbol{c}_1$, $\hat{\boldsymbol{c}}_2$, $\hat{\boldsymbol{c}}_3$, and $\hat{\boldsymbol{c}}_4$ were 0.188, 0.257, 0.266, and 0.267, respectively. These results indicate that as the layer becomes lower, the correspondence rate in attribute-specific areas becomes larger and support the qualitative observations.

References

[1] 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

[2] H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. Courville. Modulating early visual processing by language. In *NIPS*, 2017. 3

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 3

[4] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017. 2, 3

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop*, 2013. 2

[8] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010. 2

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

[10] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. In *ICML Workshop*, 2015. 2

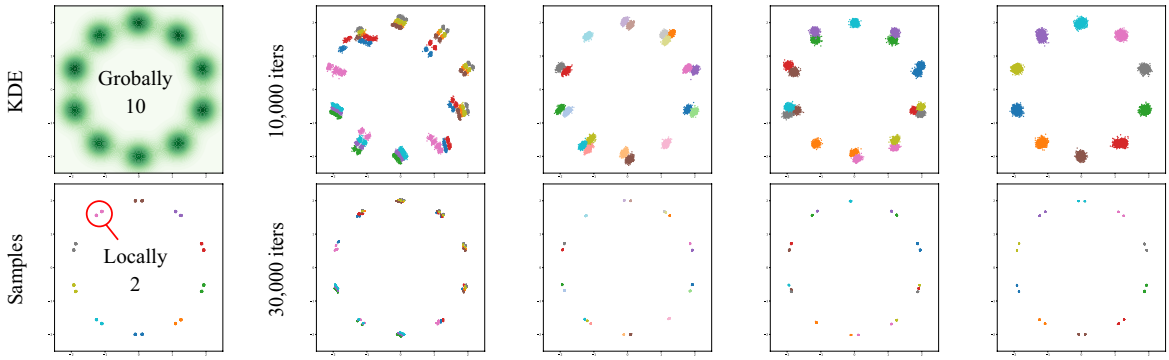


Figure 10. **Evaluation on simulated data:** (a) We used simulated data, which have globally ten categories and locally two categories. In (b), $10 \times 10 = 100$ categories are learned at the same time. In (c)(d), 20 categories are learned at the same time, causing unbalanced and non-hierarchical clustering. In (e), ten global categories are first discovered then two local categories are learned. Upper left: kernel density estimation (KDE) plots. Others: samples from real data or models. Same color indicates same $\boldsymbol{c}_1^1$ category.

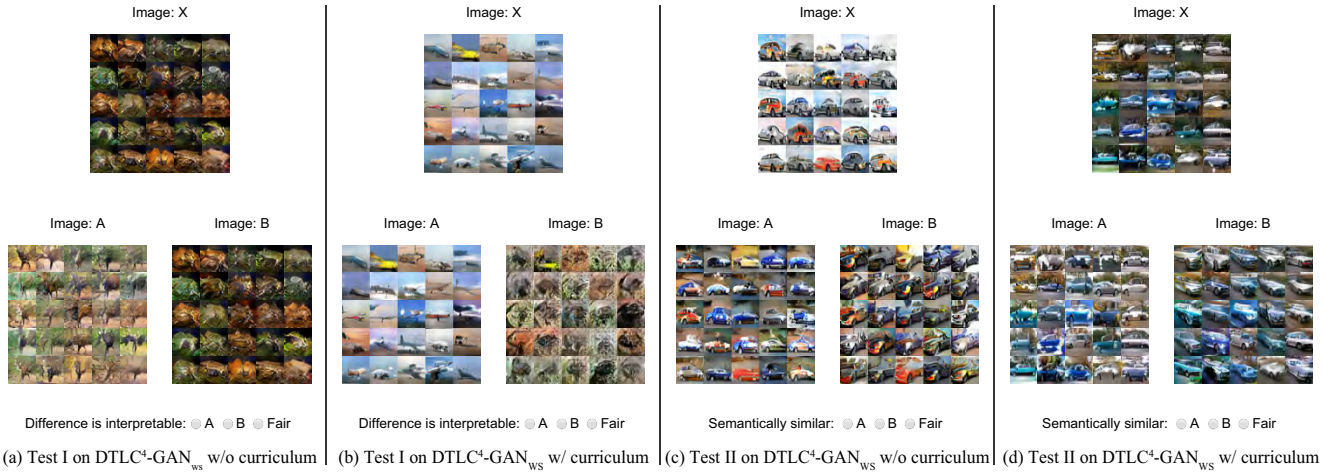


Figure 11. **User interfaces for XAB tests:** (a) Samples in "Image: A" are generated from latent variables in which one dimension of $\hat{\boldsymbol{c}}_L$ is changed. Samples in "Image: B" are generated from latent variables in which one dimension of $\boldsymbol{z}$ is changed. (b) Samples in "Image: A" are generated from latent variables in which one dimension of $\boldsymbol{z}$ is changed. Samples in "Image: B" are generated from latent variables in which one dimension of $\hat{\boldsymbol{c}}_L$ is changed. (c) Samples in "Image: A" are generated from latent variables in which $\boldsymbol{c}_4$ is varied. Samples in "Image: B" are generated from latent variables in which $\boldsymbol{c}_2$ is varied. (d) Samples in "Image: A" are generated from latent variables in which $\boldsymbol{c}_4$ is varied. Samples in "Image: B" are generated from latent variables in which $\boldsymbol{c}_2$ is varied.

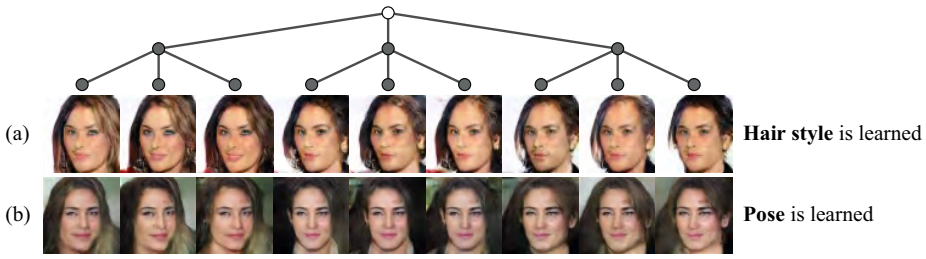


Figure 12. **Representation comparison between two models that are learned in fully unsupervised manner with different initialization:** In (a), samples are generated from one model, while, in (b), samples are generated from another model. In each row, $\boldsymbol{c}^1$ and $\boldsymbol{c}^2$ are varied per three images and per image, respectively. In this setting, learning targets (in (a), hair style and in (b), pose) depend on initialization because this dataset can be categorized in various ways.

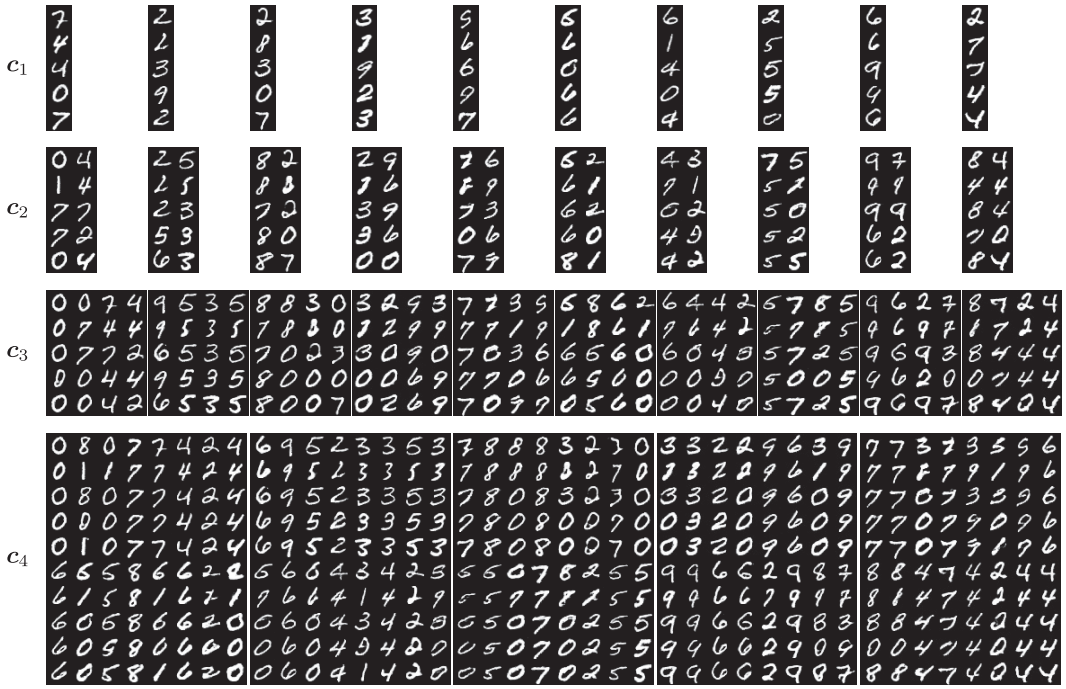


Figure 13. **Manipulating latent codes in DTLC^4-GAN (learned without curriculum) on MNIST:** In results noted with $\boldsymbol{c}_l$ $(l = 1, \cdots, 4)$, each column includes five samples generated from same $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_l$ but different $\boldsymbol{z}$ and random $\boldsymbol{c}_{l+1}, \cdots, \boldsymbol{c}_L$. In each block, each row contains samples generated from same $\boldsymbol{z}$ and $\boldsymbol{c}_1$ but different $\boldsymbol{c}_2, \cdots, \boldsymbol{c}_L$. In particular, $\boldsymbol{c}_i$ $(i = 2, \cdots, l-1)$ was varied per $\prod_{j=i}^{l-1} k_j$ images, and $\boldsymbol{c}_l$ was varied per image. $\boldsymbol{c}_i$ $(i = l+1, \cdots, L)$ was randomly chosen.

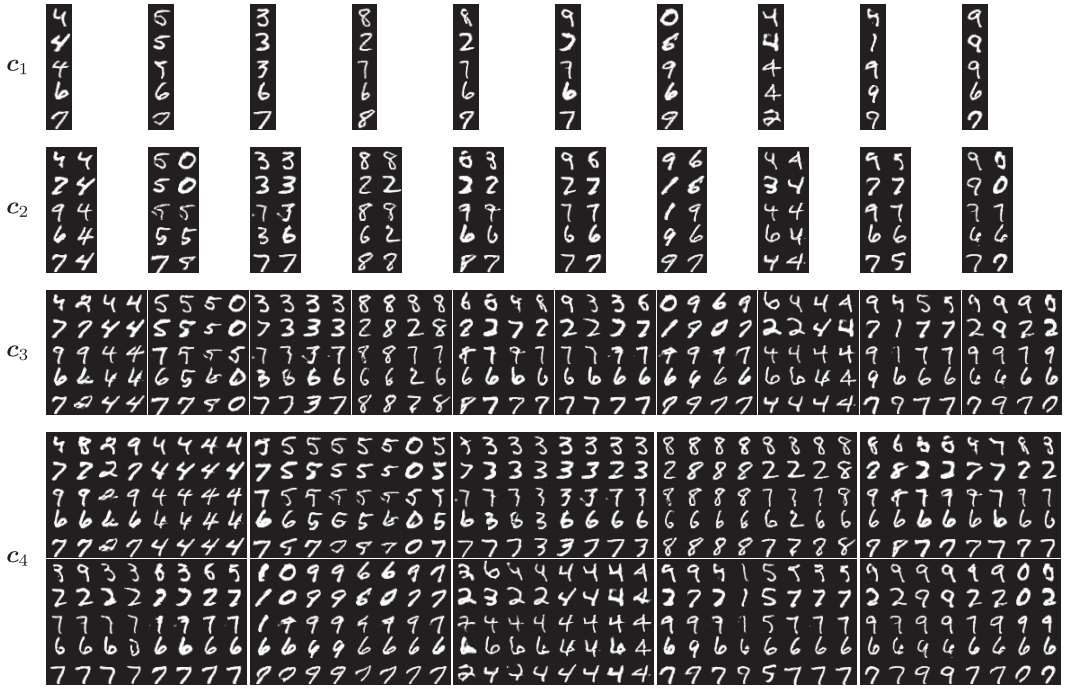


Figure 14. **Manipulating latent codes in DTLC^4-GAN (learned only with curriculum for regularization) on MNIST:** View of figure is same as that in Figure 13

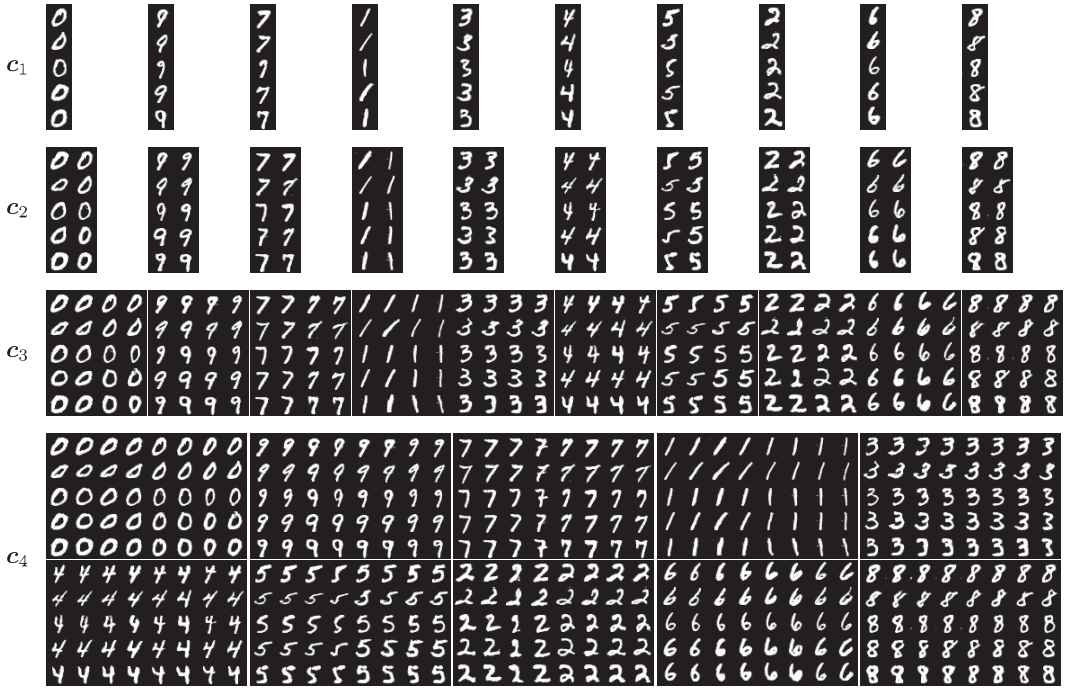


Figure 15. **Manipulating latent codes in DTLC^4-GAN (learned with full curriculum) on MNIST:** View of figure is same as that in Figure 13

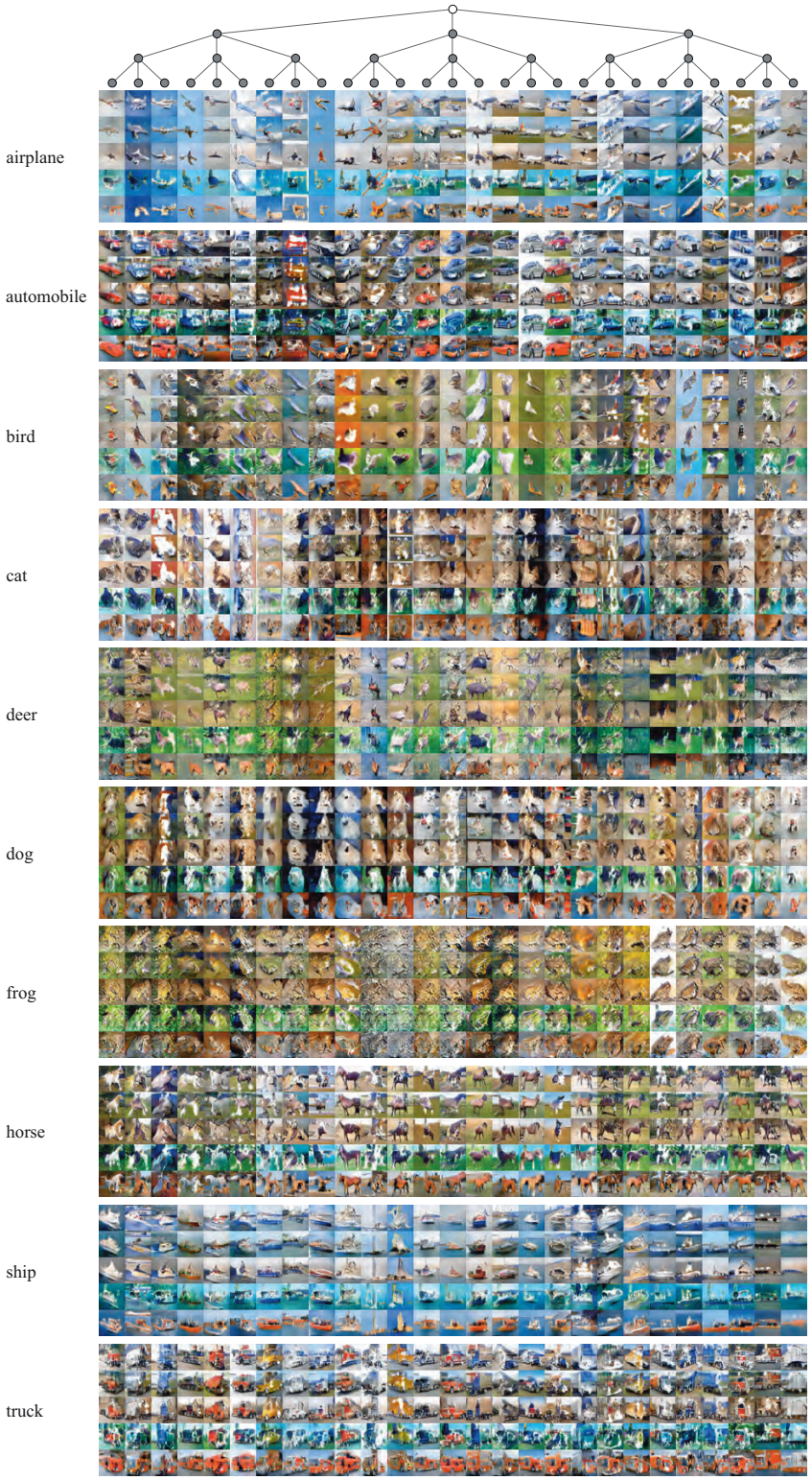


Figure 16. **Manipulating latent codes in DTLC^4-GAN$_{\mathrm{WS}}$ (learned without curriculum) on CIFAR-10:** In each block, each column includes five samples generated from same $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_4$ but different $\boldsymbol{z}$. Each row contains samples generated from same $\boldsymbol{z}$ and $\boldsymbol{c}_1$ but different $\boldsymbol{c}_2$, $\boldsymbol{c}_3$, and $\boldsymbol{c}_4$. In particular, $\boldsymbol{c}_2$, $\boldsymbol{c}_3$, and $\boldsymbol{c}_4$ were varied per nine images, per three images, and per image, respectively. Among all blocks, samples in ith row $(i = 1, \cdots, 5)$ share same $\boldsymbol{z}$.

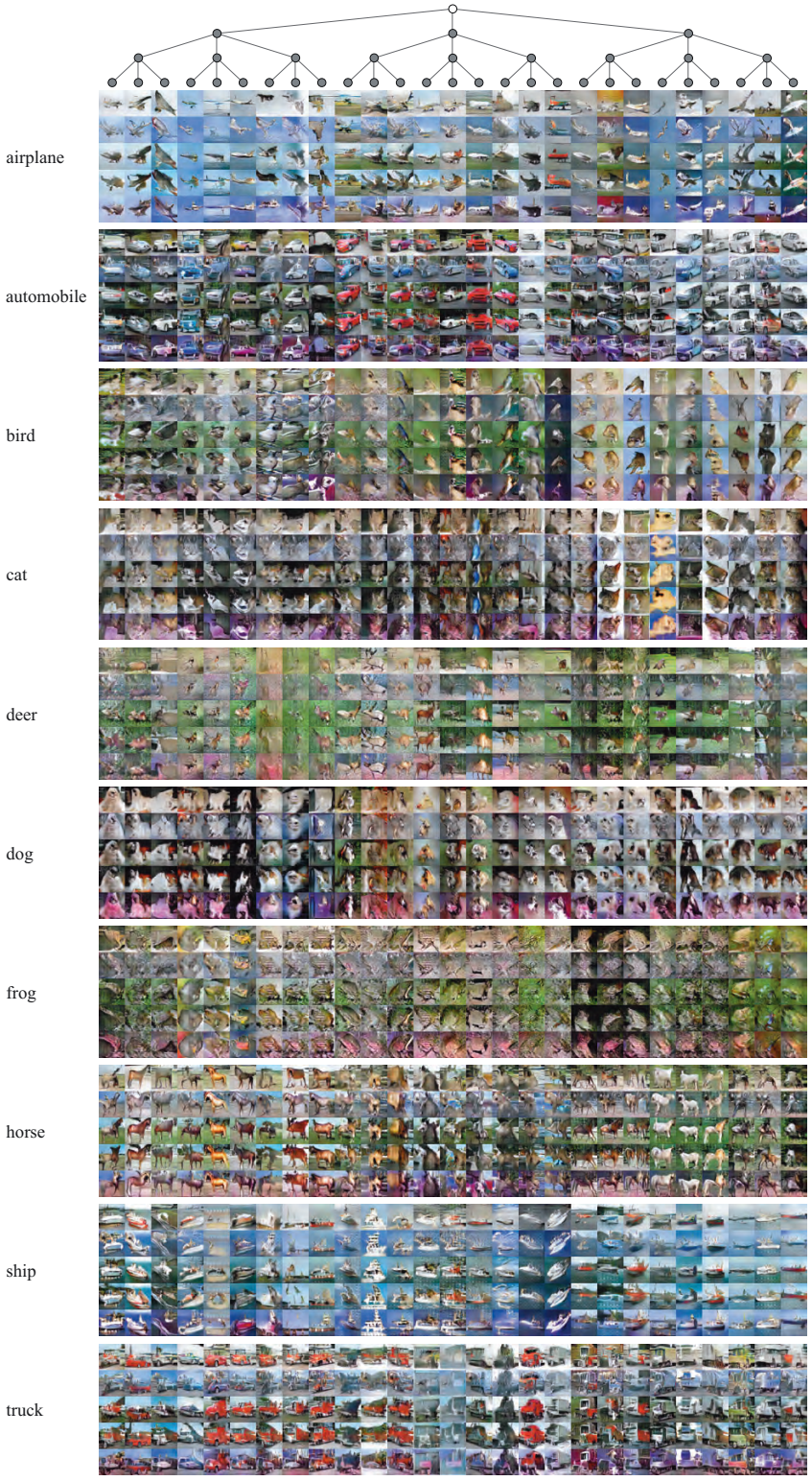


Figure 17. **Manipulating latent codes in DTLC^4-GAN$_{\text{WS}}$ (learned only with curriculum for regularization) on CIFAR-10:** View of figure is same as that in Figure 16

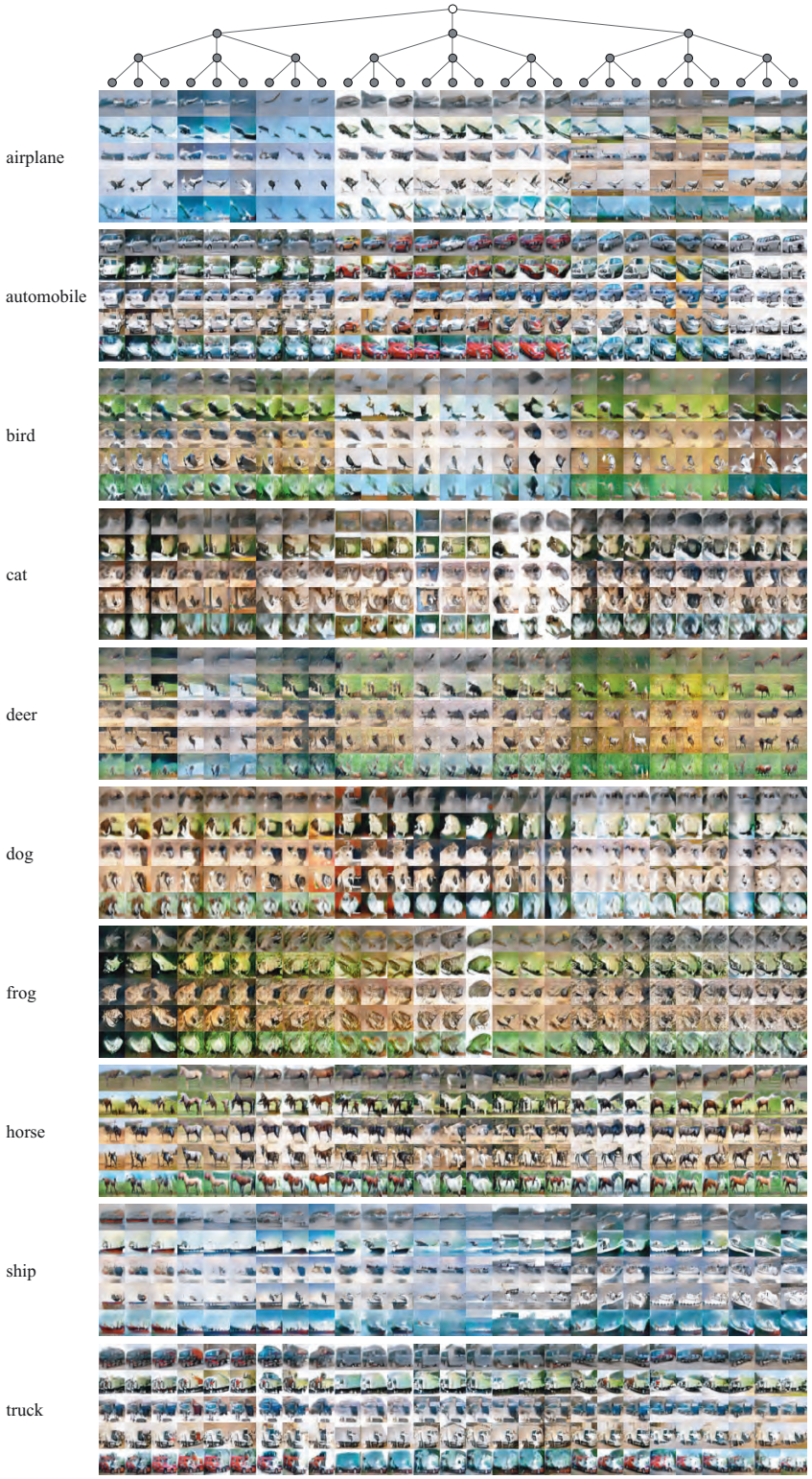


Figure 18. **Manipulating latent codes in DTLC^4-GAN$_{\text{WS}}$ (learned with full curriculum) on CIFAR-10:** View of figure is same as that in Figure 16

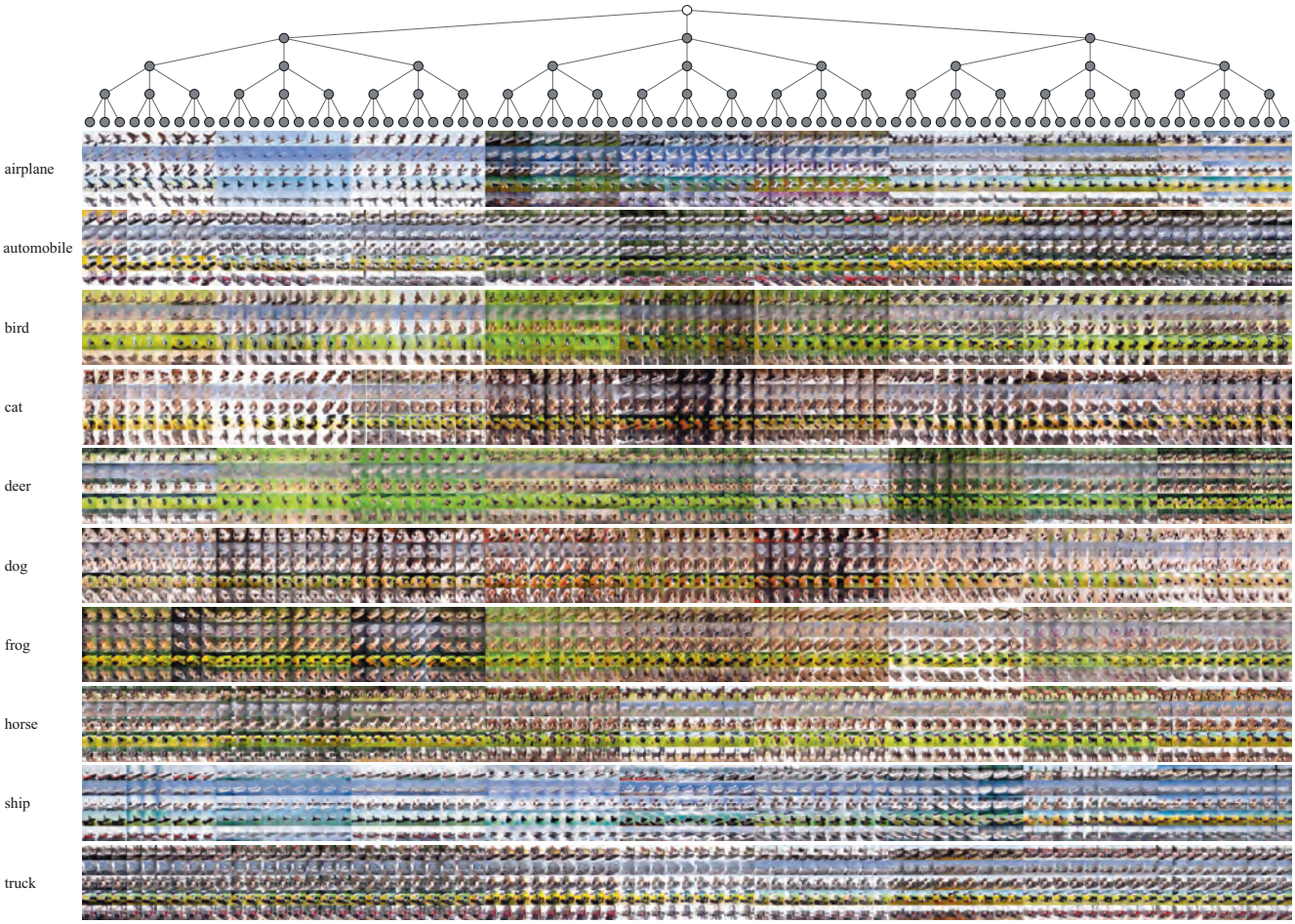


Figure 19. **Manipulating latent codes in DTLC^5-GAN$_{\mathrm{WS}}$ (learned with full curriculum) on CIFAR-10:** View of figure is similar to that in Figure 16. Total of $10 \times 3 \times 3 \times 3 \times 3 = 810$ categories were learned in weakly supervised setting.

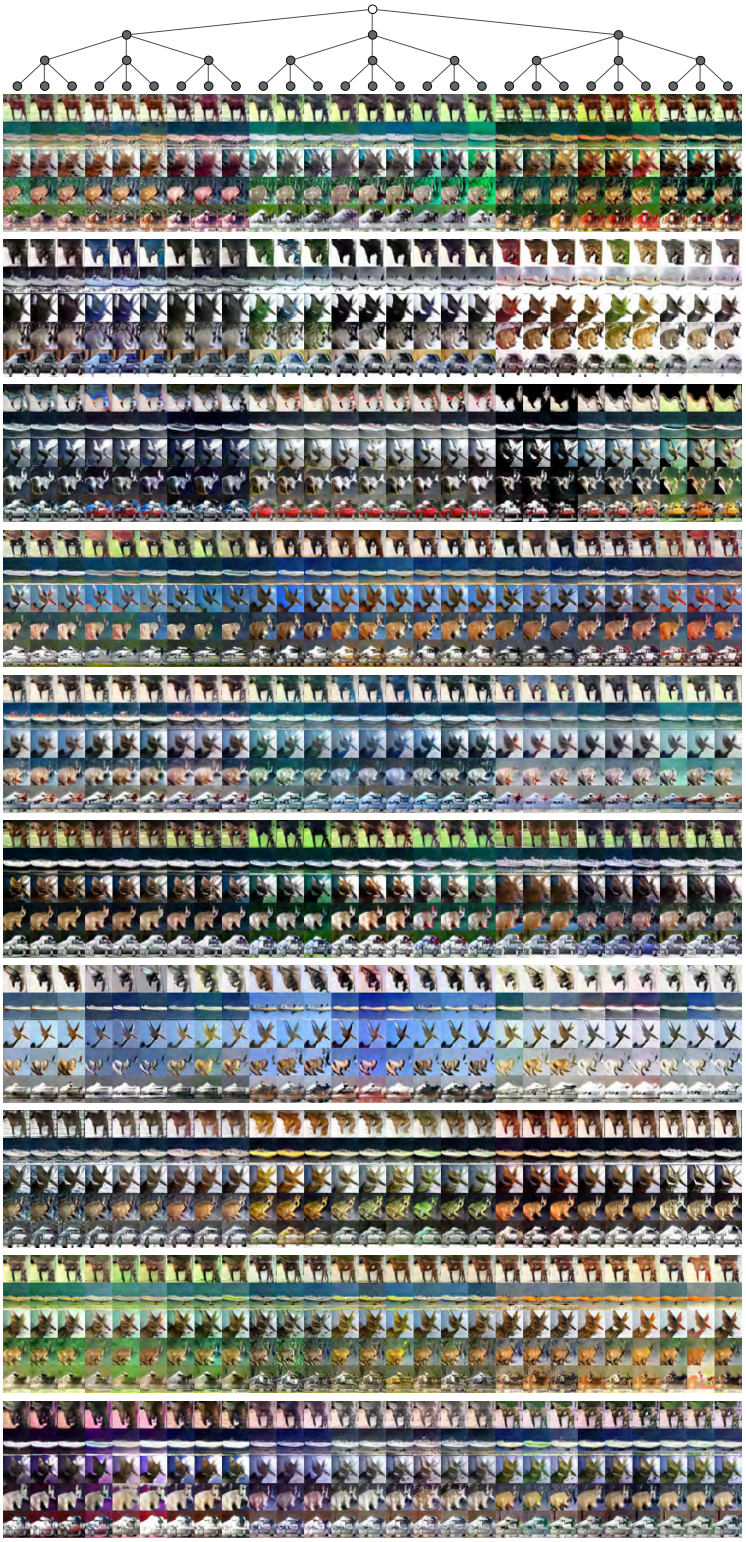


Figure 20. **Manipulating latent codes in DTLC^4-WGAN-GP on CIFAR-10:** View of figure is similar to that in Figure 16. All categories ($10 \times 3 \times 3 \times 3 = 270$) were learned in fully unsupervised setting.

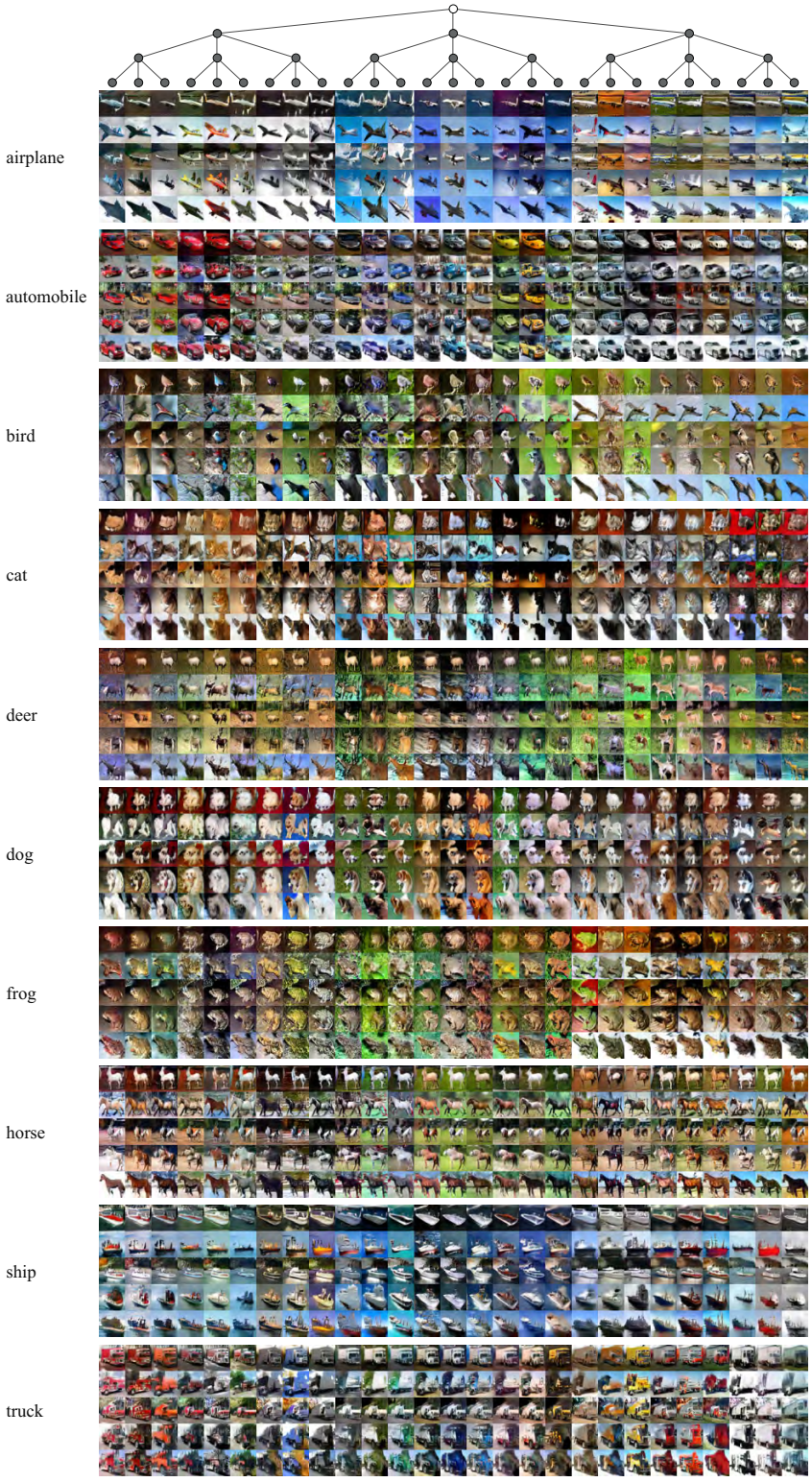


Figure 21. **Manipulating latent codes in DTLC^4-WGAN-GP$_{\mathrm{WS}}$ on CIFAR-10:** View of figure is similar to that in Figure 16. All categories ($10 \times 3 \times 3 \times 3 = 270$) were learned in weakly supervised setting.

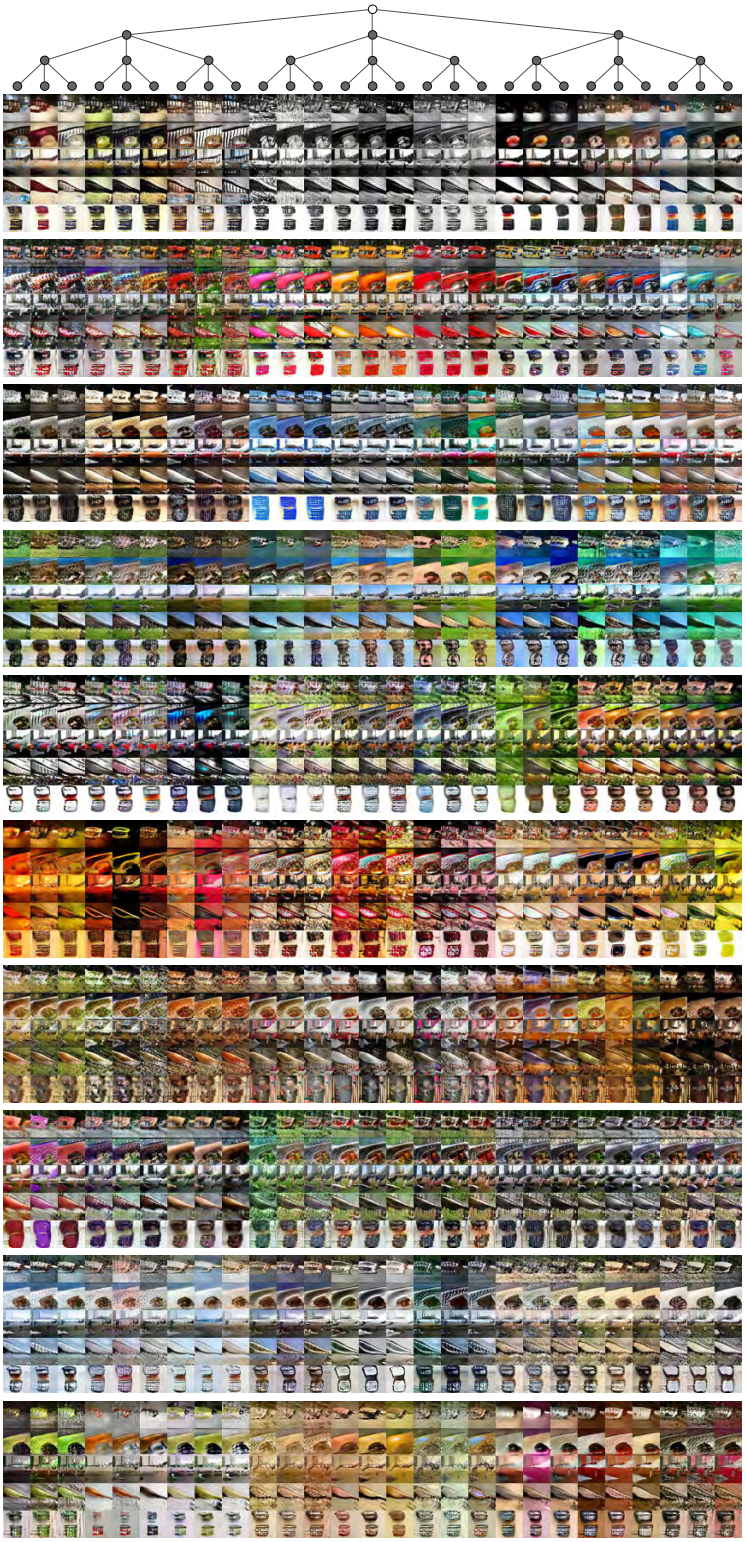


Figure 22. **Manipulating latent codes in DTLC^4-WGAN-GP on Tiny ImageNet:** View of figure is similar to that in Figure 16. All categories ($10 \times 3 \times 3 \times 3 = 270$) were learned in fully unsupervised setting.

(a-i) Varying c_1^1 (b-i) Varying c_1^2 for $c_1^{1(1)} = 1$ (c-i) Varying c_2^1 for $c_1^{1(1)} = 1$

(a-ii) Varying c_1^2 (b-ii) Varying c_1^2 for $c_1^{1(2)} = 1$ (c-ii) Varying c_2^2 for $c_1^{1(2)} = 1$

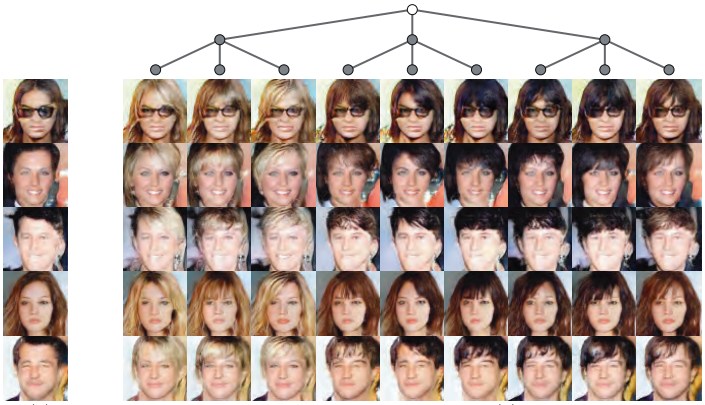
(a-iii) Varying c_1^3 (b-iii) Varying c_1^2 for $c_1^{1(3)} = 1$ (c-iii) Varying c_2^3 for $c_1^{1(3)} = 1$

(a-iv) Varying c_1^4 (b-iv) Varying c_1^2 for $c_1^{1(4)} = 1$ (c-iv) Varying c_2^4 for $c_1^{1(4)} = 1$

(a-v) Varying c_1^5 (b-v) Varying c_1^2 for $c_1^{1(5)} = 1$ (c-v) Varying c_2^5 for $c_1^{1(5)} = 1$

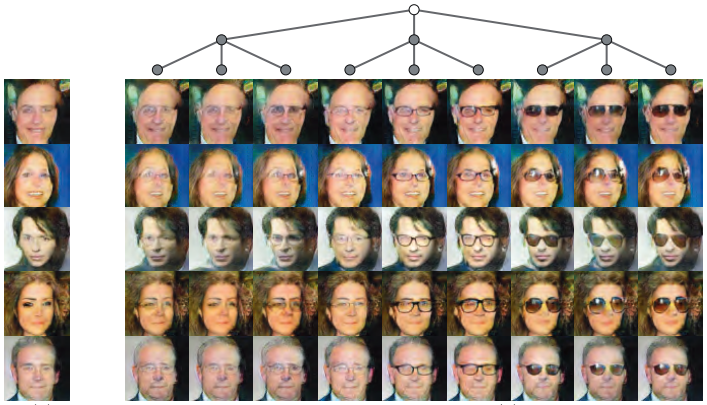
(a) InfoGAN$_{\text{C5}}$ (b) InfoGAN$_{\text{C1D1}}$ (c) DTLC²-GAN

Figure 23. **Manipulating latent codes on 3D Faces:** In each block, each column includes five samples generated from same latent codes but different noise. Each row contains samples generated from same noise and same discrete codes but different continuous codes (varied from left tor right). In tandem blocks, samples in ith row ($i = 1, \cdots, 5$) were generated from same noise. (a) In **InfoGAN**$_{\text{C5}}$, each continuous code $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_5$ captures independent and exclusive semantic features (e.g., orientation of lighting in c_1^4 and elevation in c_1^5). (b) In **InfoGAN**$_{\text{C1D1}}$, discrete code $\boldsymbol{c}_1^1$ captures pose, while continuous code $\boldsymbol{c}_1^2$ captures orientation of lighting regardless of $\boldsymbol{c}_1^1$. Also in this model, each code captures independent and exclusive semantic features. (c) In **DTLC^2-GAN**, discrete code $\boldsymbol{c}_1^1$ captures pose, and continuous codes $\boldsymbol{c}_2^1, \cdots, \boldsymbol{c}_2^5$ capture detailed variations for each pose. In this model, lower layer codes learn category-specific (in this case, pose-specific) semantic features conditioned on higher layer codes.



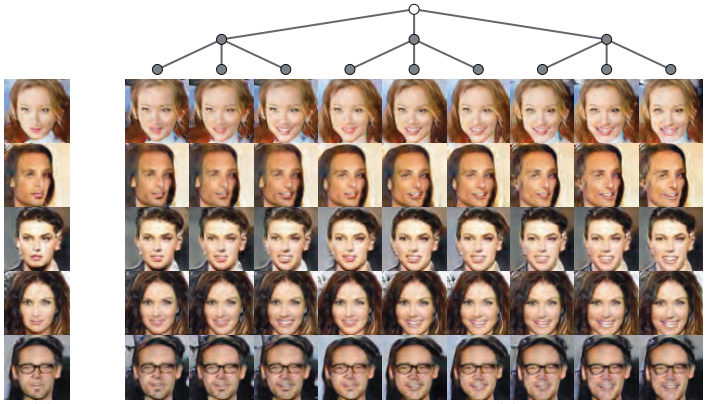
(a) $c_1^{1(1)} = 1$ (b) Varying $\boldsymbol{c}_2, \boldsymbol{c}_3$ for $c_1^{1(2)} = 1$

Figure 24. **Manipulating latent codes in DTLC^3-GAN$_{\text{WS}}$ on CelebA (bangs):** Each column includes five samples generated from same $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$ but different $\boldsymbol{z}$. In (a), samples are generated from $c_1^{1(1)} = 1$, i.e., attribute is absent. In this case, hierarchical representations are not learned. In (b), samples are generated from $c_1^{1(2)} = 1$, i.e., attribute is present. In this case, hierarchical representations are learned. In each row, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ are varied per three images and per image, respectively.



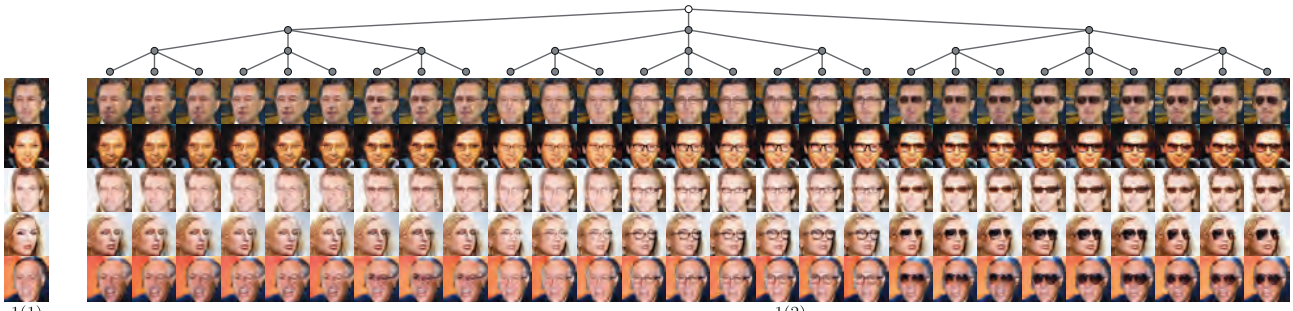
(a) $c_1^{1(1)} = 1$ (b) Varying $\boldsymbol{c}_2, \boldsymbol{c}_3$ for $c_1^{1(2)} = 1$

Figure 25. **Manipulating latent codes in DTLC^3-GAN$_{\text{WS}}$ on CelebA (glasses):** View of figure is same as that in Figure 24



(a) $c_1^{1(1)} = 1$ (b) Varying $\boldsymbol{c}_2, \boldsymbol{c}_3$ for $c_1^{1(2)} = 1$

Figure 26. **Manipulating latent codes in DTLC^3-GAN$_{\text{WS}}$ on CelebA (smiling):** View of figure is same as that in Figure 24



(a) $c_1^{1(1)} = 1$ (b) Varying $\boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4$ for $c_1^{1(2)} = 1$

Figure 27. **Manipulating latent codes in DTLC^4-GAN$_{\text{WS}}$ on CelebA (glasses):** View of figure is similar to that in Figure 24. Total of $1 + 1 \times 3 \times 3 \times 3 = 28$ categories were learned in weakly supervised settings.

Figure 28. **Hierarchical image retrieval using DTLC^4-GAN$_{\text{WS}}$ on CelebA (glasses)**