Disentangling Disentangled Representations: Towards Improved Latent Units via Diffusion Models (Supplementary Material)

Contents

A. Additional Materials

An overview of the paper and a brief presentation video are available on our project page: https://youngjun-jun.github.io/dis-dis-rep.

B. Datasets

This section provides an overview of the benchmark datasets (Fig. 1). The number of samples for each dataset is shown in Table 1.

Cars3D [15]. The Cars3D dataset is a car dataset created from CAD models with the following ground truth factors: elevation, azimuth, object type.

Shapes3D [2]. The Shapes3D dataset is composed of 3D shapes with the following ground truth factors: floor hue, wall hue, object hue, scale, shape, orientation.

MPI3D-toy [6]. The MPI3D-toy dataset is part of the MPI3D dataset created to benchmark representation learning in simulated and real-world environments. It focuses on the toy type with the following ground truth factors: object color, object shape, object size, camera height, background color, first DOF, second DOF.

CelebA [13]. The CelebFaces Attributes Dataset (CelebA) is a face attributes dataset with 40 attributes. Although CelebA is not specifically designed for disentanglement, it allows for the validation of effects in real-world scenarios using various attributes.

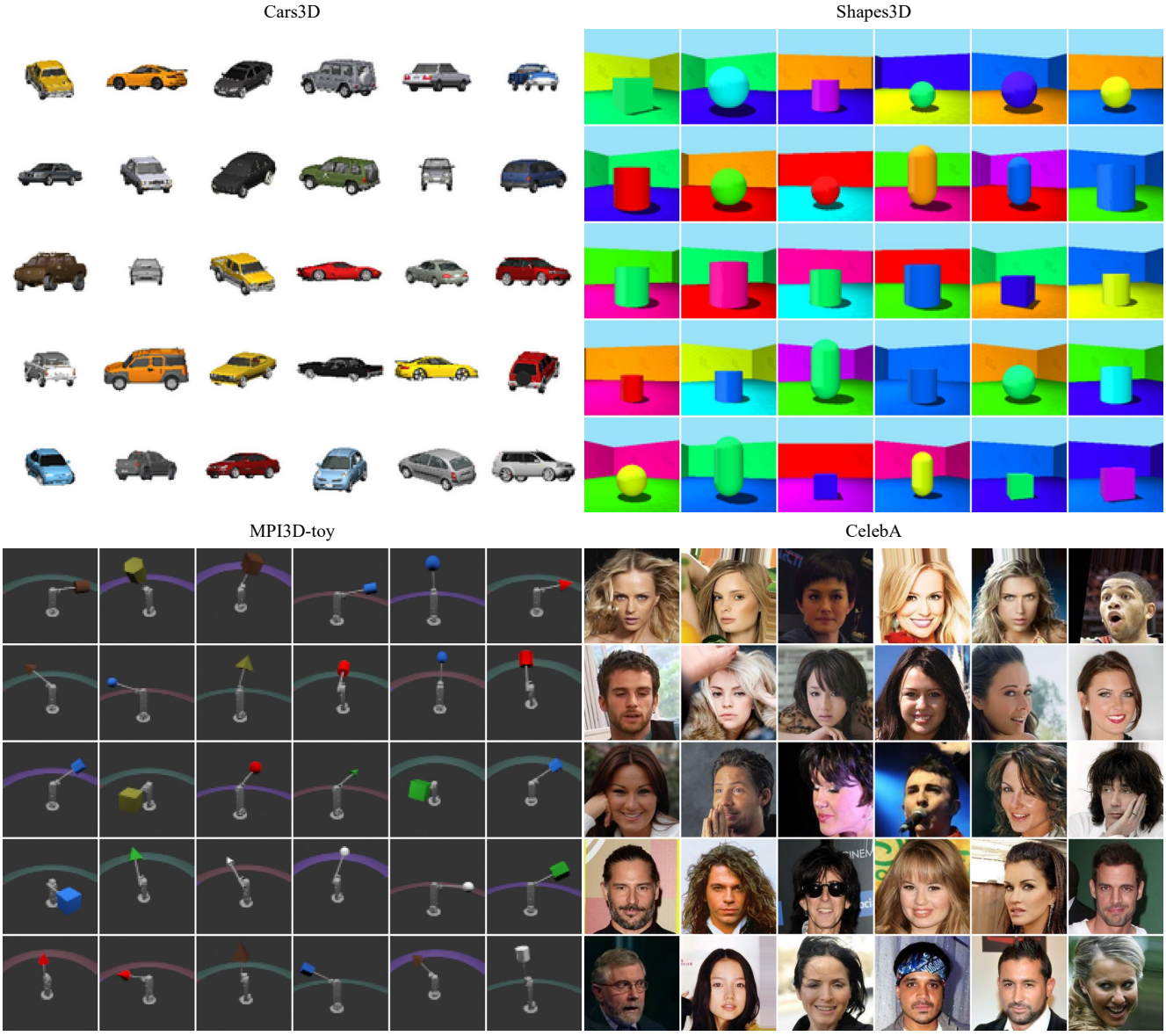


Figure 1. The samples from each dataset

Table 1. The number of samples for each dataset

Cars3D	Shapes3D	MPI3D	CelebA
17,568	480,000	1,036,000	202,599

C. More Implementation Details

C.1. Training Details

C.1.1 Algorithms

Entire training framework. As in Algorithm 1, the LDM training process using DyGA is conducted on an epoch-by-epoch basis. After every r epochs, *anchor selection* is performed for *feature alignment* in the subsequent r epochs. Considering the dataset size, we set $r = 5$ for the Cars3D dataset, while $r = 1$ was used for the others.

Anchor selection in DyGA. The anchor selection process (Algorithm 2) consists of four stages. First, subspaces are updated using HDDC. Then, a Gaussian to split is selected based on density, and the two split Gaussians are updated using only the features within the cluster. Next, excessively small Gaussians that could act as outliers during feature alignment are removed. Finally, the Gaussians are updated using HDDC. These steps are performed at the latent unit level, with multi-processing utilized for each latent unit and accelerated through matrix multiplication. The anchor selection process for a single latent unit is completed within 30 seconds.

Feature alignment in DyGA. Naïvely aligning features with the anchors selected through anchor selection hinders the backpropagation process. Therefore, as shown in Algorithm 3, we use the Gumbel-softmax function [9] with a sufficiently small $\tau = 0.0001$ to prevent distortion of the Gaussian mean while still allowing back-propagation. After this, the features undergo alignment in the direction of the anchor.

Algorithm 1 Entire training framework

Input: $\mathbf{X}$: dataset, $\{\sigma_t\}_{t=1}^{T}$: Noise schedule, $E(\cdot)$: VQ-encoder, $\epsilon_\theta(\cdot)$: Denoising network, $f(\cdot)$: Feature extractor
for $epoch = 1, \ldots, max\_epoch$ **do**
  for $\mathbf{x}_0^i \in \mathbf{X}$ **do**
    Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $t \sim \text{Uniform}(\{0, 1, \ldots, T\})$
    $\mathbf{z}_t^i \leftarrow E(\mathbf{x}_0^i) + \sigma_t \epsilon$
    $\mathbf{c}_i \leftarrow f(\mathbf{x}_0^i)$
    if $epoch \geq r$ **then** $\tilde{\mathbf{c}}_i \leftarrow Feature\_alignment(\mathbf{c}_i)$
    else $\tilde{\mathbf{c}}_i \leftarrow \mathbf{c}_i$
    end if
    Predict noise $\epsilon_\theta(\mathbf{z}_t, t, \tilde{\mathbf{c}}_i)$
    Compute loss and gradient
    Update parameters θ
    Anchor_selection
  end for
end for

Algorithm 2 Anchor selection of the k^{th} cluster

Input: $\mathbf{c} \in \mathbb{R}^d$: latent units
Output: μ_k, Σ_k, w_k, Λ_k, $\mathbf{v}_k$, D_k
Initialize μ_k, Σ_k, w_k, Λ_k, $\mathbf{v}_k$, D_k
for $iter = 1, \ldots, max\_iter$ **do**
  E-step: Calculate r_{ik}
  M-step: Update μ_k, Σ_k, w_k
  Update subspaces: Compute Λ_k, $\mathbf{v}_k$, D_k
end for
Calculate density of $clusters \in \mathcal{F}$
while $\mathcal{F} \neq \emptyset$ **do**
  Choose cluster $\mathbf{f} \in \mathcal{F}$
  Split cluster $\mathbf{f}$
  E-step, **M-step**, and **Update subspaces** in the $cluster$
  Update $\mathcal{F}$
end while
Remove small Gaussians
for $iter = 1, \ldots, max\_iter$ **do**
  E-step, **M-step**, and **Update subspaces**
end for

Algorithm 3 Feature alignment

Input: $\mathbf{c} \in \mathbb{R}^d$: latent unit, τ: softmax parameter, λ: feature alignment parameter
Output: $\hat{\mathbf{c}} \in \mathbb{R}^d$
E-step: Calculate responsibility r
Gumbel-Softmax:

1. Generate Gumbel noise g_k
2. Compute $y_k = \text{softmax}\left(\frac{r_k + g_k}{\tau}\right)$

Update $\mu_k \leftarrow \sum_k y_k \mu_k$
Compute multiplier $\delta \leftarrow \lambda \exp\left(-\frac{1}{d}\sum_{j=1}^{d}\left|\frac{\mathbf{c}_i^j - \boldsymbol{\mu}_k^j}{\mathbf{c}^j}\right|\right)$
Align latent unit $\tilde{\mathbf{c}} \leftarrow \mathbf{c} + \delta(\boldsymbol{\mu}_k - \mathbf{c})$

C.1.2 LDM training details

During the training process of LDM, we set the batch size to 512 for the Cars3D [15], Shapes3D [2], and MPI3D-toy [6] datasets, and 64 for the CelebA dataset. The learning rate and Exponential Moving Average (EMA) rate were set to 0.0001 and 0.9999, respectively, for all datasets, following EncDiff [22].

C.2. Model Architecture

The latent diffusion model (LDM) [17] we used is a diffusion model for images in the latent space reduced by VQ-GAN [5]. The architectures of VQ-GAN and the LDM denoising U-Net are shown in Table 2 and Table 3, respectively. Additionally, the feature extractor (Table 4), similar to previous studies [21, 22], uses a structure where a scalar value is taken for each latent unit through a CNN module and then passed through independent MLP modules. We used the same settings for all datasets.

Table 2. VQ-GAN architecture parameters

Parameter	Value
Embedding dimensionality	3
Number of embeddings	2048
Channels in first conv layer	32
Channel multipliers	[1, 2, 4]
Residual blocks per layer	2
Dropout rate	0.0
Discriminator start epoch	0
Discriminator loss weight	0.75
Codebook loss weight	1.0

Table 3. Denoising U-Net architecture parameters

Parameter	Value
Input image size	16
Input channels	3
Base channels	64
Attention resolutions	[1, 2, 4]
Residual blocks per layer	2
Channel multipliers	[1, 2, 4, 4]
Attention heads	8
Scale-shift normalization	True
Context dimension	32
Dropout rate	0.1
Skip dropout rate	0.2
Noise schedule	Linear
Diffusion timestep	1000

Table 4. Feature extractor architecture

CNN module
Conv $7 \times 7 \times 3 \times 64$, stride$=1$, padding$=3$
BatchNorm
ReLU
Conv $4 \times 4 \times 64 \times 128$, stride$=1$, padding$=3$
BatchNorm
ReLU
Conv $4 \times 4 \times 128 \times 256$, stride$=2$, padding$=1$
BatchNorm
ReLU
Conv $4 \times 4 \times 256 \times 256$, stride$=2$, padding$=1$
BatchNorm
ReLU
Conv $4 \times 4 \times 256 \times 256$, stride$=2$, padding$=1$
BatchNorm
ReLU
FC 4096×4096
ReLU
FC 4096×256
ReLU
FC $256 \times N$
MLP module
FC 1×256
ReLU
FC 256×512
ReLU
FC 512×32
ReLU

D. More Experiments

D.1. Results for more metrics

In this subsection, as shown in Table 5, we evaluate the disentanglement performance of our method using not only the FactorVAE score [10] and DCI disentanglement [4], but also additional metrics such as MIG [3], Modularity score [16], SAP score [12], and InfoMEC InfoM score [7].

Table 5. Results for more metrics on Cars3D, Shapes3D and MPI3D-toy datasets

Dataset	FactorVAE score↑	DCI↑	MIG↑	Modularity score↑	SAP score↑	InfoM score↑
Cars3D	0.941 ± 0.002	0.414 ± 0.013	0.109 ± 0.002	0.934 ± 0.001	0.009 ± 0.002	0.417 ± 0.004
Shapes3D	1.000 ± 0.000	0.938 ± 0.001	0.507 ± 0.002	0.930 ± 0.001	0.183 ± 0.006	0.569 ± 0.002
MPI3D-toy	0.930 ± 0.004	0.627 ± 0.002	0.364 ± 0.001	0.882 ± 0.002	0.174 ± 0.002	0.495 ± 0.003

Table 6. Comparison between DyGA and FSQ

Method	FactorVAE score↑	DCI↑
Baseline	0.856 ± 0.004	0.586 ± 0.002
+FSQ, SD	0.606 ± 0.006	0.384 ± 0.002
+DyGA, SD	$\mathbf{0.930} \pm 0.004$	$\mathbf{0.627} \pm 0.002$

D.2. DyGA and SD Analysis

DyGA Analysis. Our proposed *Dynamic Gaussian Anchoring* (DyGA) can dynamically adjust the position and number of anchors. A similar approach can be found in the quantization methods used in variational autoencoder (VAE) [11]-based approaches [7, 8]. One such method, finite scalar quantization (FSQ) [14], can be applied to diffusion-based models as a way to organize latent units. However, when we set the number of codebooks to 1 and the levels to [8, 5, 5], performance actually degraded, as shown in Table 6, with similar performance drops observed in other settings. This suggests that attempts to organize latent units in diffusion-based models using a fixed number of quantization values or fixed quantized values may not be suitable. Therefore, we introduced DyGA, which dynamically adjusts the number and positions of anchors, resulting in significant performance improvements.

Table 7. Skip and Backbone Dropout

Method	FactorVAE score↑	DCI↑
+BD-0.1	0.798 ± 0.005	0.610 ± 0.002
Baseline	0.856 ± 0.004	0.586 ± 0.002
+SD-0.1	0.852 ± 0.005	0.598 ± 0.007
+SD-0.2	$\mathbf{0.863} \pm 0.005$	$\mathbf{0.615} \pm 0.002$
+SD-0.3	0.855 ± 0.006	0.603 ± 0.002

SD Analysis. *Skip Dropout* (SD) stochastically blinds skip connection features to prevent the continuous accumulation of factor information in specific weights. This forces the diffusion U-Net to rely on the backbone features connected to the feature extractor, which continuously provide factor information. To analyze this effect more deeply, we examine the impact of SD by using backbone dropout (BD), which dropouts backbone features in parts with skip connections. On the other hand, it has been shown that dropout rates between 0.4 and 0.8 do not affect performance improvement [19], and we have experimentally found that a high skip dropout ratio (*e.g.*, $1 - p = 1.0$) negatively impacts the convergence of the diffusion model. Therefore, we used skip dropout rates of 0.1, 0.2, and 0.3, while the backbone dropout rate was set to 0.1. As a result, as shown in Table 7, an appropriate skip dropout rate was effective, whereas backbone dropout degraded performance.

D.3. Latent Interpolation

There has been no exploration of the disentangled representation space of diffusion-based models (*i.e.*, the space of outputs from the feature extractor). In this subsection, we visualize images generated by interpolating latent units extracted from two images. This intermediate image generation enables image morphing [1, 20, 23, 24], which shows the transformation between images as a video. Experimentally, when using spherical linear interpolation (slerp) [18] for the diffusion latent interpolation method in DiffMorpher [23], the generated intermediate images appeared unnatural. Instead, as shown in Fig. 2, images generated using linear interpolation suggest the potential for image morphing through disentanglement.

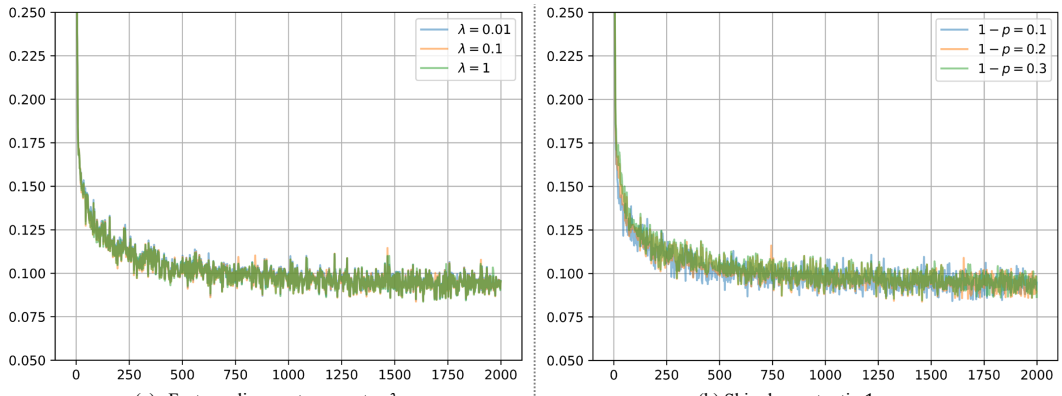
(a) CelebA (b) Cars3D

Figure 2. Visualization of latent interpolation on the Cars3D and CelebA datasets. **(a)** For CelebA, we observe natural transitions between two images in terms of `hair color`, `hair style`, `skin color`, `background color`, `gender`, and `smile`. **(b)** Similarly, in Cars3D, we observe smooth changes in `vehicle type`, `color`, `azimuth`, and `elevation`.

E. More Visualizations

E.1. Training loss curve

In this subsection, we plot the training loss curve according to different parameters. From Fig. 3, we can confirm that our methods remain stable despite changes in training parameters (λ: feature alignment parameter, $1-p$: skip dropout ratio).



(a) Feature alignment parameter λ (b) Skip dropout ratio $1-p$

Figure 3. Training loss curve

E.2. More Latent Interchange Results

One way to verify if a trained feature extractor extracts well-disentangled representations is to manipulate them directly. If the visualized results accurately reflect the intended changes in representation, it indicates that the latent units faithfully represent the factors. Here, we refer to changing one of the latent units of the source image to the latent unit of the target image as a latent interchange. The images generated using the latent units created through latent interchange conditionally alter the source image using the single latent unit information of the target image. Fig. 4 visualizes how well the feature extractor is trained in each dataset through latent interchange. Our method demonstrates that the images generated using latent interchange effectively reflect a single characteristic of the target image.

E.3. More Attention Map Visualizations

Since our diffusion model receives conditions through cross attention, it is possible to visualize the attention map. This attention map shows which areas of the image are being highlighted, indicating which parts of the image each latent unit uses to generate. The results on various datasets can be seen in Figs. 5 and 6.

E.4. More Latent Unit Visualizations

Disentangled representation learning aims to make each latent unit sensitive to a single fundamental factor while being invariant to the other factors. At this point, the latent unit with the highest association (*e.g.*, normalized mutual information) to a given factor best reflects the information of that factor. Therefore, we visualize the latent units for various data points in the Shapes3D dataset (Fig. 7) to see how well the latent units separate the factor attributes.

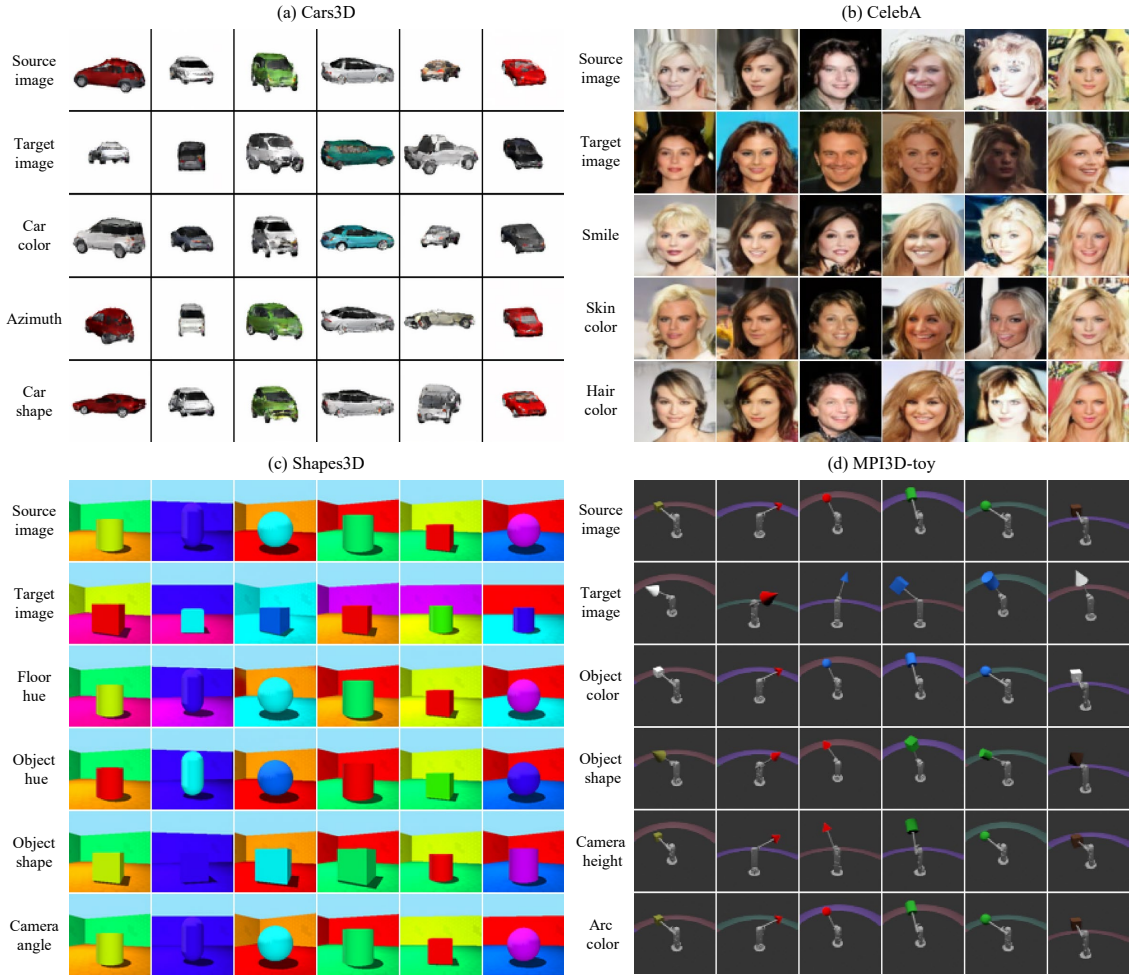


Figure 4. Latent interchange results on Cars3D, Shapes3D, MPI3D-toy, and CelebA datasets

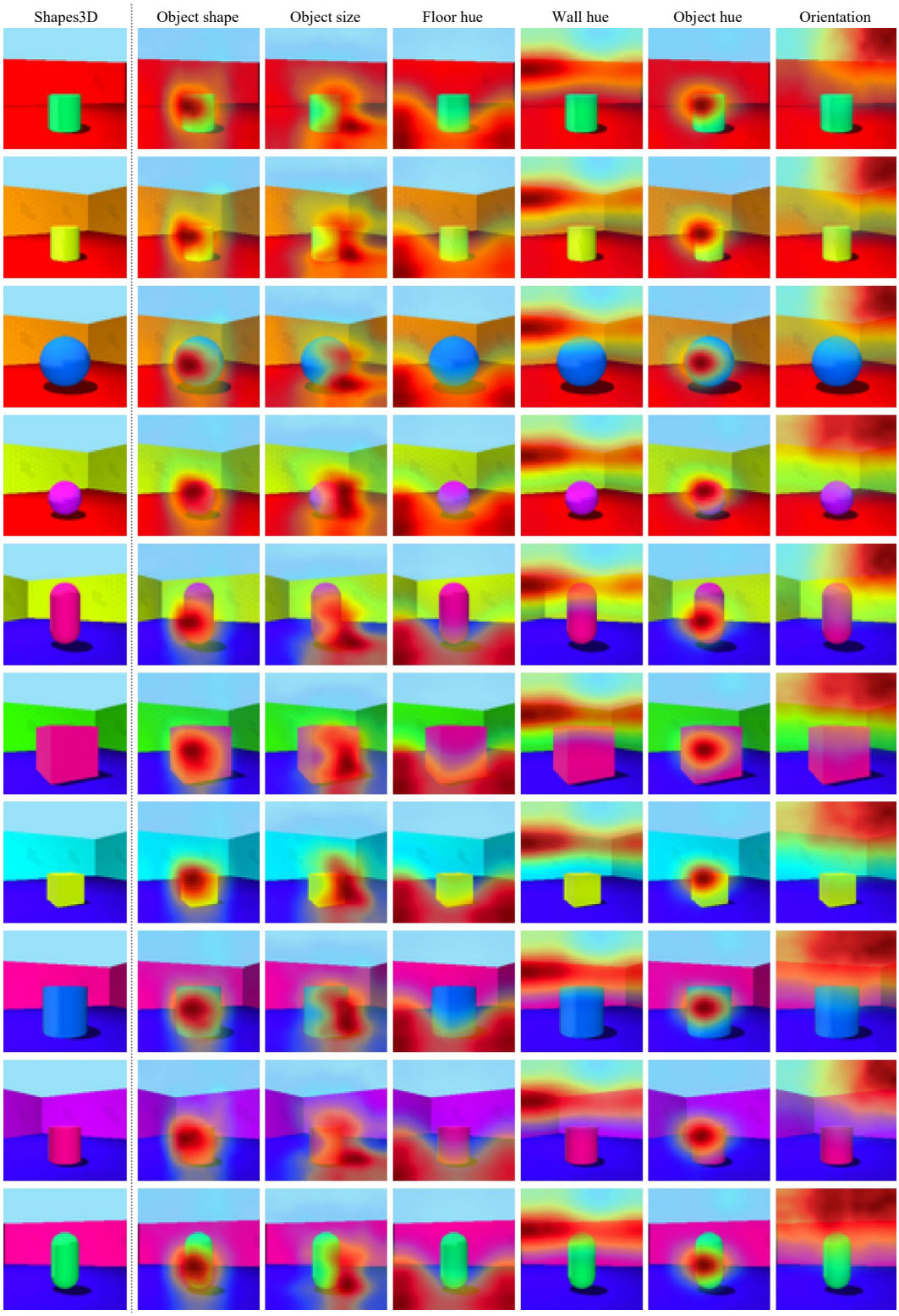


Figure 5. Attention map visualizations on Shapes3D dataset

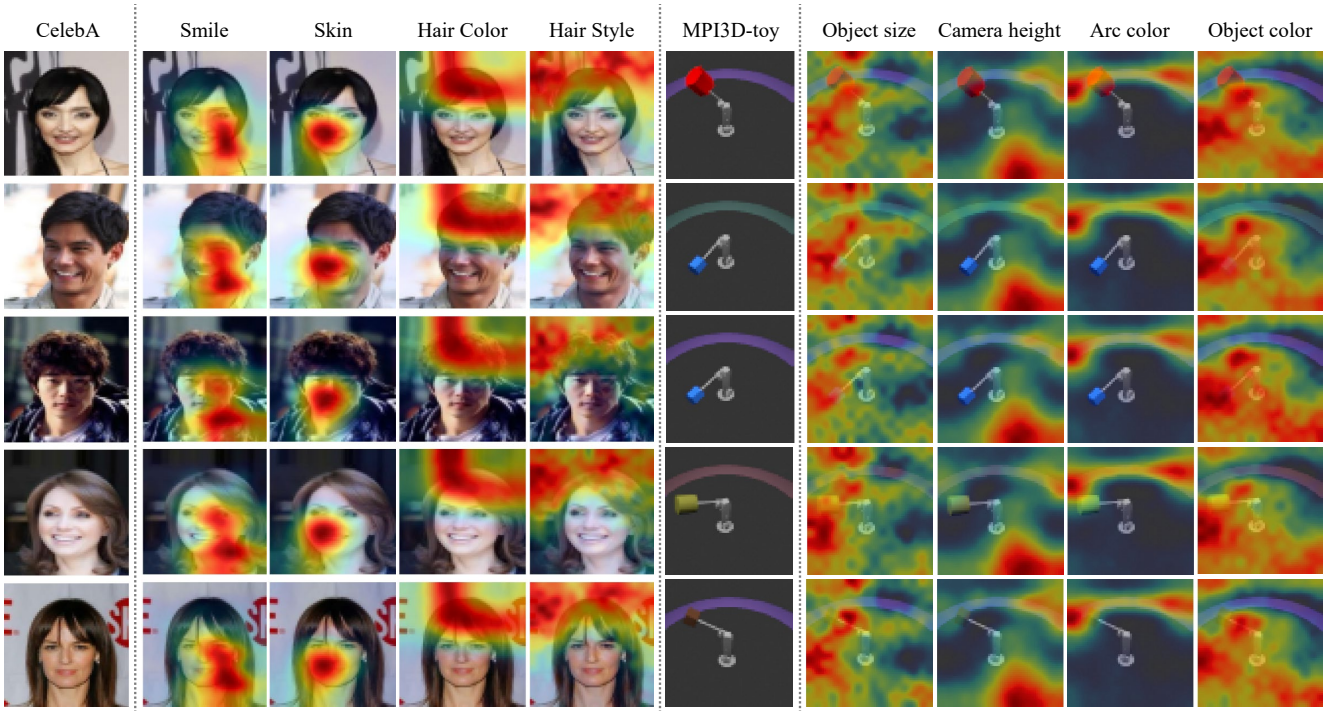


Figure 6. Attention map visualizations on CelebA and MPI3D-toy datasets

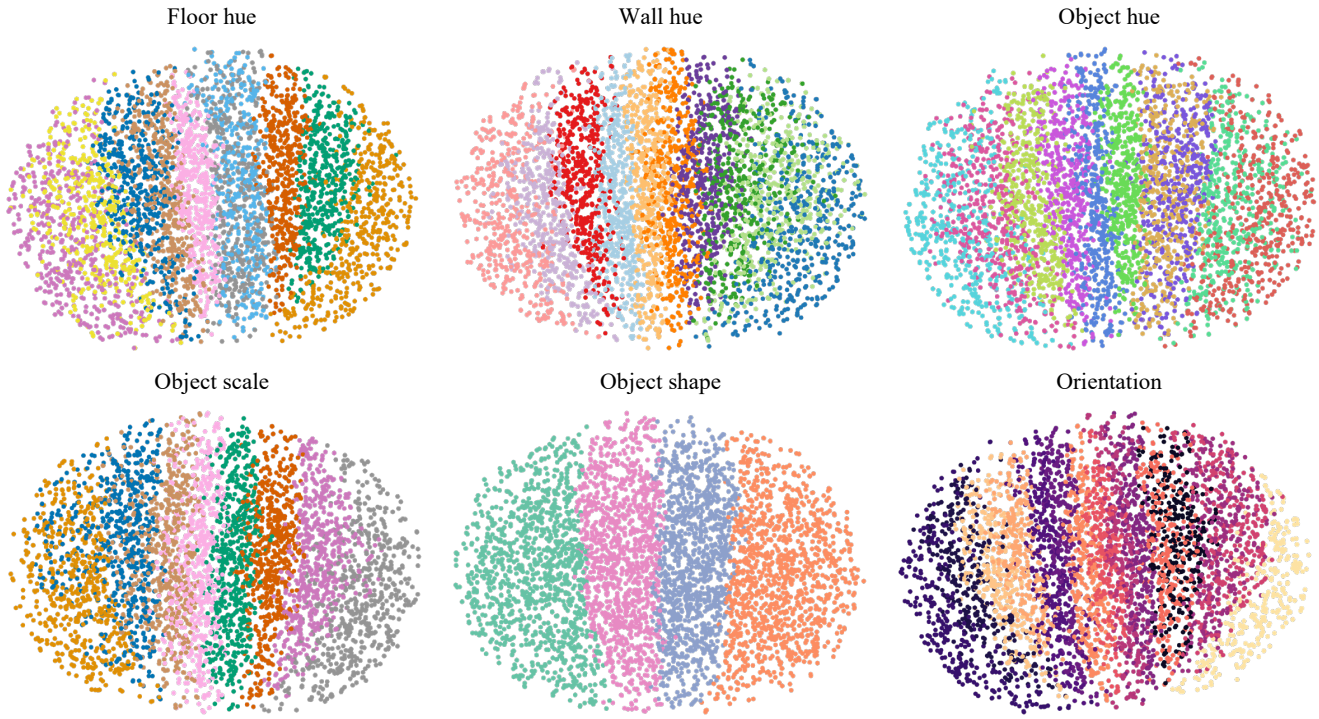


Figure 7. Visualization of latent units on Shapes3D dataset. Each visualized latent unit here is associated with the written factor (*e.g.*, it has the highest normalized mutual information). Each color represents an attribute of the factor, such as `red` in object color or `cube` in object shape.

References

[1] Alyaa Aloraibi. Image morphing techniques: A review. *Technium: Romanian Journal of Applied Sciences and Technology*, 9:41–53, 2023. 5

[2] Chris Burgess and Hyunjik Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018. 1, 4

[3] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018. 4

[4] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *International conference on learning representations*, 2018. 4

[5] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 4

[6] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 4

[7] Kyle Hsu, William Dorrell, James Whittington, Jiajun Wu, and Chelsea Finn. Disentanglement via latent quantization. *Advances in Neural Information Processing Systems*, 36, 2024. 4, 5

[8] Kyle Hsu, Jubayer Ibn Hamid, Kaylee Burns, Chelsea Finn, and Jiajun Wu. Tripod: Three complementary inductive biases for disentangled representation learning. *arXiv preprint arXiv:2404.10282*, 2024. 5

[9] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 3

[10] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International conference on machine learning*, pages 2649–2658. PMLR, 2018. 4

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[12] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017. 4

[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. 2

[14] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023. 5

[15] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. *Advances in neural information processing systems*, 28, 2015. 1, 4

[16] Karl Ridgeway and Michael C Mozer. Learning deep disentangled embeddings with the f-statistic loss. *Advances in neural information processing systems*, 31, 2018. 4

[17] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 4

[18] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985. 5

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 5

[20] George Wolberg. Image morphing: a survey. *The Visual Computer*, 14:360–372, 1998. 5

[21] Ancong Wu and Wei-Shi Zheng. Factorized diffusion autoencoder for unsupervised disentangled representation learning. 2024. 4

[22] Tao Yang, Cuiling Lan, Yan Lu, et al. Diffusion model with cross attention as an inductive bias for disentanglement. *arXiv preprint arXiv:2402.09712*, 2024. 4

[23] Kaiwen Zhang, Yifan Zhou, Xudong Xu, Bo Dai, and Xingang Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2024. 5

[24] Bhushan Zope and Soniya B. Zope. A survey of morphing techniques. *International Journal of Advanced engineering, Management and Science*, 3:81–87, 2017. 5