Diffusion Models with Grouped Latents for Interpretable Latent Space

Sangyun Lee [1] **Gayoung Lee** [2] **Hyunsu Kim** [2] **Junho Kim** [2] **Youngjung Uh** [3]

Abstract

Latent variable models are useful tools for discovering independent generative factors of data without human supervision. From an ODE formulation, diffusion models are invertible latent variable models, but unlike other models like VAEs, their latent variables are often not interpretable. For example, traversing a single element of the latent noise does not lead to a meaningful variation of generated contents. To settle this issue, we propose to divide a latent vector into multiple groups of elements and design different noise schedules for each group. By doing so, we can allow each group to control only certain elements of data, explicitly giving interpretable meaning. Applying our method in the frequency domain, the latent variable becomes a hierarchical representation where individual groups encode data at different levels of abstraction. We show several applications of such representation including disentanglement of semantic attributes or image editing.

1. Introduction

Discovering underlying generative factors of data without human supervision is an important problem in machine learning. In representation learning, it is often assumed that data $\boldsymbol{x}$ is sampled from $\int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$, where $\boldsymbol{z}$ denotes underlying data-generating factors with the factorized marginal distribution. The aim of latent variable models (Fischer & Igel, 2012; Goodfellow et al., 2014; Kingma & Welling, 2013) is to approximate the true generative process $p(\boldsymbol{x}|\boldsymbol{z})$ by training $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ such that $p_{\boldsymbol{\theta}}(\boldsymbol{x}) \approx p(\boldsymbol{x})$. By estimating $p(\boldsymbol{x}|\boldsymbol{z})$, we can uncover the independent generative factors of data using its posterior $p(\boldsymbol{z}|\boldsymbol{x})$, which can be useful for downstream tasks that require the various factors of data. However, since generative learning itself is extremely difficult, the latent variable models have not been so successful in generating large-scale datasets.

Figure 1: Synthesis results of previous diffusion model by traversing a single element of a latent vector. The results of the previous model are almost identical.

Recently, diffusion models (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2020; Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021) have achieved remarkable success on large-scale datasets. Diffusion models are trained by learning the drift of a generative ODE that yields the same marginal distributions as a given forward diffusion process. In such a viewpoint, diffusion models learn a one-to-one mapping between $\boldsymbol{x}$ and $\boldsymbol{z}$ defined via ODE, hence the representation is easily obtainable by solving generative ODE backward without requiring additional inference networks.

However, unlike other models like VAEs, it is difficult to infer the meaning of each latent element of diffusion models since varying it does not lead to any meaningful variation of generated contents. The reason might be the absence of compact latent space. In other latent variable models, it is possible to set the latent dimensionality close to the intrinsic dimension of data, therefore forcing the majority of variables to capture meaningful generative factors. Yet, it is nontrivial to adapt this into diffusion models as $\boldsymbol{x}$ and $\boldsymbol{z}$ have the same dimensionality.

To this end, we propose to divide a latent vector into multiple groups of elements and design different noise schedules for each group. It allows each group to control only certain elements of data, explicitly giving interpretable meaning. Applying our method in the frequency domain, the latent variable becomes a hierarchical representation where individual groups encode data at different levels of abstraction. We show several applications of such representation including disentanglement of semantic attributes or image editing.

[1]Soongsil University [2]NAVER AI Lab [3]Yonsei University. Correspondence to: Junho Kim <jhkim.ai@navercorp.com>, Youngjung Uh <yj.uh@yonsei.ac.kr>.

Accepted to ICML workshop on Structured Probabilistic Inference Generative Modeling

2. Background on Diffusion Models

Diffusion models are generative models that synthesize data by simulating a reverse-time SDE of a given diffusion process, which is often converted to the marginal-preserving ODE for efficient sampling. From a rectified flow (Liu et al., 2022; Liu, 2022) (or similarly, stochastic interpolant (Albergo et al., 2023; Albergo & Vanden-Eijnden, 2022)) perspective, the forward diffusion process for variance-preserving diffusion models (Song et al., 2020) can be viewed as a nonlinear interpolation between data $\boldsymbol{x} \sim p(\boldsymbol{x})$ and noise $\boldsymbol{z} \sim p(\boldsymbol{z})$:

$$\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}) = \alpha(t)\boldsymbol{x} + \sqrt{1-\alpha(t)^2}\boldsymbol{z}, \tag{1}$$

where $\alpha(t)$ is an nonlinear function of t with $a(0) = 1$ and $a(1) \approx 0$, and $p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. Some recent work (Lipman et al., 2022; Liu et al., 2022) instead use the linear interpolation

$$\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}) = (1-t)\boldsymbol{x} + t\boldsymbol{z} \tag{2}$$

for a constant velocity that has better sampling-time efficiency.

Diffusion models are trained by minimizing a weighted denoising autoencoder loss (Vincent, 2011)

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim U(0,1)}\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}[\lambda(t)||\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{\theta}}(\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}), t)||_2^2] \tag{3}$$

with a weighting function $\lambda(t)$. Instead of predicting $\boldsymbol{x}$, Liu et al. (2022) directly train a vector field $\mathbf{v}_\theta(\boldsymbol{x}_t, t)$ to match the time derivative $\frac{\partial \boldsymbol{x}_t}{\partial t}$ of Eq. (2) by optimizing

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim U(0,1)}\mathbb{E}_{\boldsymbol{x},\boldsymbol{z}}[||(\boldsymbol{z} - \boldsymbol{x}) - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}), t)||_2^2]. \tag{4}$$

Inference and sampling are done by solving the following ODE (Liu et al., 2022) forward and backward in time, respectively:

$$d\boldsymbol{z}_t = \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{z}_t, t)dt, \tag{5}$$

where dt is an infinitesimal timestep.

Remark 2.1. *In optima of Eq. (4), Eq. (5) maps between $p(\boldsymbol{x})$ and $p(\boldsymbol{z})$. See Theorem 3.3 in Liu et al. (2022) for the proof. As a consequence, the aggregate inference distribution has zero Total Correlation (TC) in the optima if we define $p(\boldsymbol{z})$ as a factorized distribution.*

3. Grouped Latent for Interpretable Representation

In practice, the representation obtained by solving Eq. (5) does not capture interesting factors of variation, as shown in Fig. 1. We attribute the reason to the absence of a well-organized latent space and aim to solve this problem by dividing a latent vector $\boldsymbol{z}$ into several groups and designing per-group noise schedules to assign a dedicated meaning to each group. More details follow.

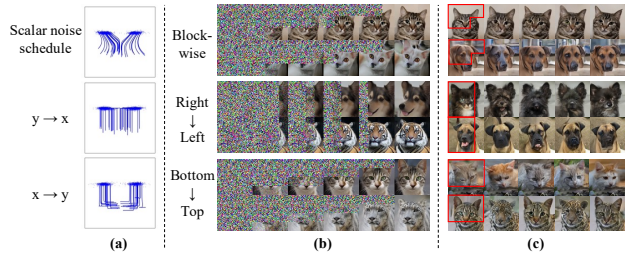


Figure 2: Generative process on (a) 2D Gaussian mixture and (b) AFHQ 64 × 64 datasets. While the previous method with scalar noise schedule denoises all elements simultaneously, our model generates each element (a) or group of pixels (b) sequentially. Varying each latent group only affects the corresponding group of pixels (indicated by red color) while others remain unchanged (c).

3.1. Per-group noise scheduling

For interpretable latent space, we divide a latent variable into k ($1 \le k \le d$) groups and assign different noise schedules for each group. Specifically, for $\boldsymbol{z} \in \mathbb{R}^d$, we divide the indices $\{1, ..., d\}$ into a partition $\{S_j\}_{j=1}^k$, and let the latent variables of each set of indices S_j participate to a certain phase, i.e., a sequence of consecutive timesteps, of the generative process.

For that, it is necessary to define separate noise schedules for each group. Lee et al. (2022) introduces several generalizations of previous diffusion models, including the utilization of a matrix-valued function $\mathbf{A}(t)$ instead of a scalar-valued function $\alpha(t)$ for the coefficient of the interpolation between data and noise. Using this, we extend Eq. (2) to

$$\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}) = \mathbf{A}(t)\boldsymbol{x} + (\mathbf{I} - \mathbf{A}(t))\boldsymbol{z}, \tag{6}$$

where $\mathbf{A}(t)$ is a diagonal matrix satisfying $\mathbf{A}(0) = \mathbf{I}$ and $\mathbf{A}(1) \approx \mathbf{0}$. Note that we build upon Eq. (2) for convenience, and other interpolations such as Eq. (1) are equally applicable. Training objective becomes

$$\min_{\boldsymbol{\theta}} \mathbb{E}[||\mathbf{A}'(t)(\boldsymbol{x} - \boldsymbol{z}) - \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{x}_t(\boldsymbol{x}, \boldsymbol{z}), \mathbf{A}(t))||_2^2], \tag{7}$$

where $\mathbf{A}'(t) = \frac{\partial \mathbf{A}(t)}{\partial t}$. Note that now $\boldsymbol{v}_\theta(\cdot)$ receives $\mathbf{A}(t)$ instead of t, which is concatenated to noised data $\boldsymbol{z}_t$.

This formulation allows us to define a different noise schedule for each element of data, therefore providing more flexibility in designing diffusion models. For $i \in S_j$, the i-th diagonal entry of $\mathbf{A}(t)$ is defined as follows:

$$\mathbf{A}(t)_{ii} = \begin{cases} 1, & (0 \le t < t_{\text{start}_j}) \\ \frac{1}{t_{\text{start}_j} - t_{\text{end}_j}} t - \frac{1}{t_{\text{start}_j} - t_{\text{end}_j}} t_{\text{end}_j}, & (t_{\text{start}_j} \le t < t_{\text{end}_j}) \\ 0, & (t > t_{\text{end}_j}) \end{cases} \tag{8}$$

Figure 3: Visualization of generative processes of previous diffusion model with scalar noise schedule and our method (frequency domain). Different from the previous scalar noise schedule, our model generates images sequentially from low frequencies to high frequencies.

Here, the elements of j-th latent group are diffused into noise during the interval $[t_{\text{start}_j}, t_{\text{end}_j}]$. The interval of each group is predefined such that they do not overlap with each other. That is, only one group of elements is diffused (and therefore generated) within a certain time interval. Since each latent group contributes to a certain phase only, the role of each group is explicitly predetermined by setting S_j, t_{start_j}, and t_{end_j}. For example, we can make diffusion models generate data in a sequential manner, as shown in Fig. 2 (a) and (b). As a result of the proposed generative process, varying each latent group only affects a specific region of the generated image, as shown in Fig. 2 (c).

3.2. Extension to frequency domain

Our generative process can be extended to the frequency domain, where the components of each frequency band are sequentially generated. This is enabled by another generalization proposed in Lee et al. (2022), the choice of coordinate systems where diffusion is performed. Using this, Eq. (6) is further generalized to

$$\bar{\boldsymbol{x}}_t(\bar{\boldsymbol{x}}, \bar{\boldsymbol{z}}) = \mathbf{A}(t)\bar{\boldsymbol{x}} + (\mathbf{I} - \mathbf{A}(t))\bar{\boldsymbol{z}}, \tag{9}$$

where $\bar{\boldsymbol{x}} = \mathbf{U}^T\boldsymbol{x}$ and $\bar{\boldsymbol{z}} = \mathbf{U}^T\boldsymbol{z}$ for an orthogonal matrix $\mathbf{U}$. Depending on the choice of basis $\mathbf{U}$, diffusion models can be extended to the frequency domain. Lee et al. (2022) uses a frequency basis $\tilde{\mathbf{U}}$ that satisfies $\mathbf{W} = \tilde{\mathbf{U}}\mathbf{D}\tilde{\mathbf{U}}^T$ where $\mathbf{W}$ is a Gaussian blurring matrix. The training loss and generative ODE remain the same, but training and inference are performed in pixel space rather than in the frequency domain for better performance.

By extending our method to the frequency domain, we can now divide an image into k frequency bands and assign each latent group to each band. Therefore, we can obtain representation organized into k-level hierarchy, where each group corresponds to a certain level of abstraction. Since different tasks in the real world require an understanding of the world at different levels of abstraction, this is a notable advantage compared to discriminative approaches (Chen et al., 2020; Grill et al., 2020), which only learn a feature with a certain level of abstraction specified by the choice of augmentations.

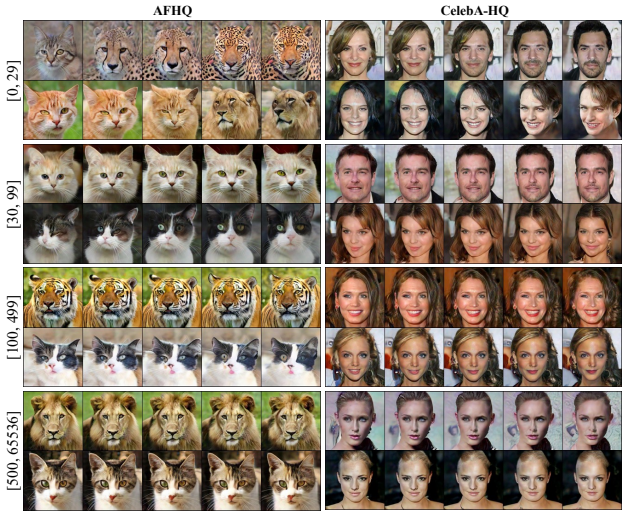


Figure 4: Generated images by interpolating the latent group for each frequency band while others fixed.

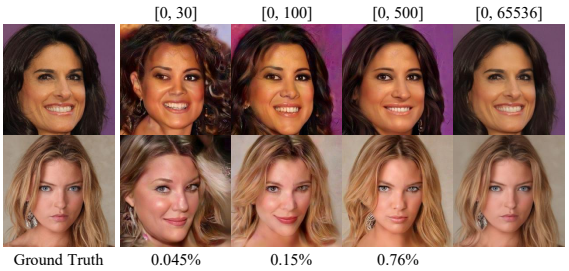


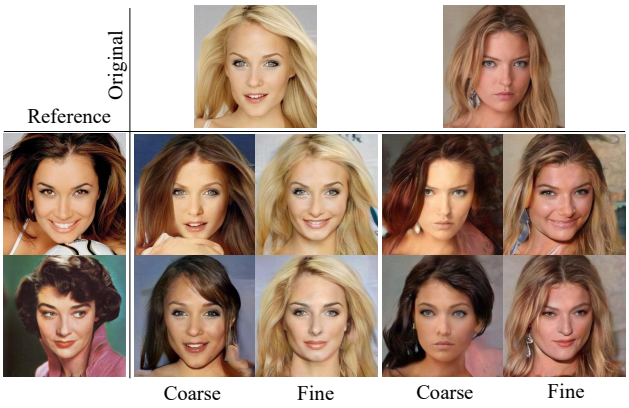
Figure 5: Reconstruction results using a subset of latent variables.

4. Experiments

In our experiment, we divide a latent vector into 4 groups (i.e. $k = 4$) in the frequency domain. Specifically, for the dataset of $r \times r$ resolution, we set $S_1 = \{0, ..., 29\}, S_2 = \{30, ..., 99\}, S_3 = \{100, ..., 499\}$, and $S_4 = \{500, ..., r^2 - 1\}$, where the indices are sorted in ascending order starting from the lowest frequency band. We set t_{start_j} and t_{end_j} such that images are generated sequentially from low frequencies to high frequencies. See Fig. 3 for visualization of generative processes. We provide additional experiments in Appendix. C.

4.1. The role of each latent group

Fig. 4 shows the images synthesized by interpolating the elements of latent vectors corresponding to each frequency band on AFHQ 256 $\times$ 256 and CelebA-HQ 256 $\times$ 256 datasets. When interpolating the coarsest frequency band [0, 29], high-level attributes such as gender, azimuth, or animal



(a) CelebA-HQ

(b) CelebA-HQ + AFHQ

Figure 6: Synthesized images by mixing the latent code of two images. We replace the latent code of *Original* with that of *Reference*. The images can be either from the same dataset (a) or different datasets (b). In (a), *coarse* means the frequency band of [0, 29], and *fine* means [100, 499]. In (b), we replace the coarse band only.

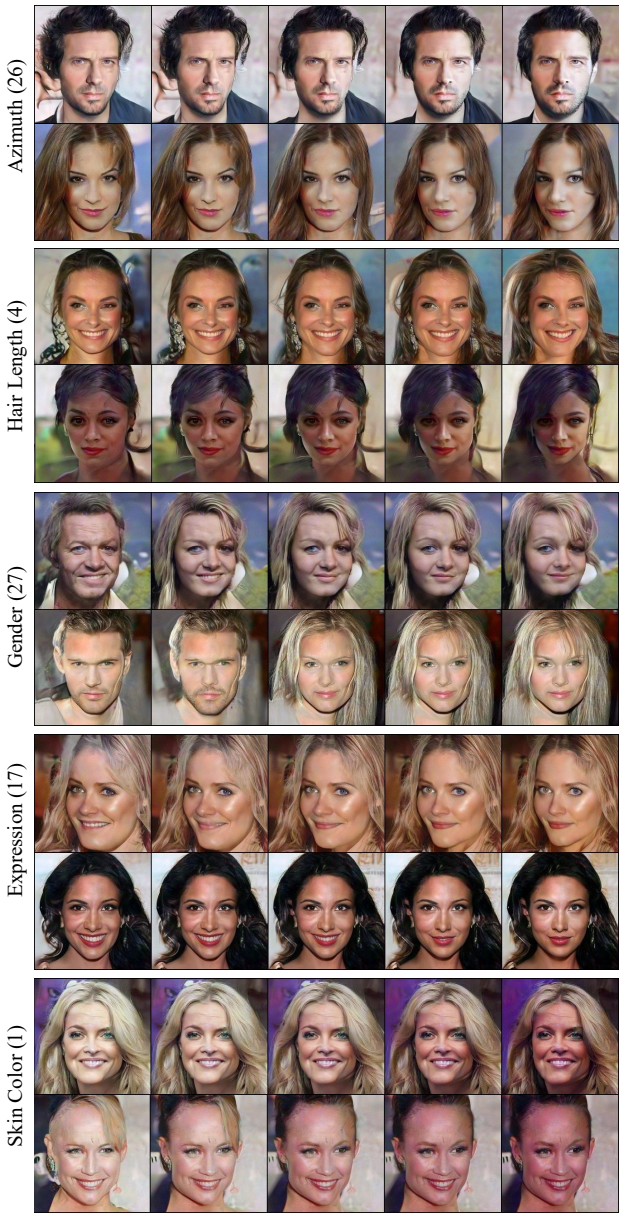


Figure 7: Synthesis results of our diffusion model, traversing a single latent variable corresponding to the lowest frequency band over $[-3, 3]$ range. The numbers in parentheses denote the indices of the element traversed.

class are transformed smoothly. Conversely, interpolating the elements of [100, 499] band results in variation in fine attributes like facial expression. We can see that the elements in [500, 65536] do not change the content of images in a meaningful way, indicating that these elements control only minor attributes in images.

As shown in Fig. 5, the latent variables in [0, 500] band are sufficient to faithfully reconstruct an image. That is, our method can effectively compress images by encoding semantic information in the coarse latent variables, which accounts for only 0.76% of the total number of elements.

These results show that our model effectively learns a latent representation with a multi-level hierarchy, each of which could be useful to tackle a variety of tasks that requires a different level of abstraction.

4.2. Image mixing

Since the inference of diffusion models is invertible, we can apply our method to editing real images. Fig. 6 shows that we can mix the style of two images by swapping their latent code of certain frequency bands. As shown in (a), the high-level attributes are transferred when we swap the coarse latent group $[0, 29]$ while swapping the fine latent group $[100, 499]$ changes attributes like facial expression. (b) shows that we can also mix images from different

datasets, in a similar fashion to DDIB (Su et al., 2022). For that, we train our model on both datasets, encode images from two datasets, and then swap latent codes in the $[0, 29]$ range. Unlike DDIB, we can selectively transfer latent codes of a certain level of abstraction since our latent variables are organized as such. This is not possible in DDIB, where latent variables have no structure and therefore possess no dedicated meaning.

4.3. Factors of variation

Recall that since $p(\bar{\boldsymbol{z}}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, the aggregate inference distribution has zero TC in the optima (see Remark. 2.1). Importantly, only a few elements in $\bar{\boldsymbol{z}}$ capture the most information in images since signals are heavily concentrated in low-frequency components. We find that such a compact, independent feature effectively encodes high-level attributes like azimuth, hair length, gender, and so on. Fig. 7 shows that manipulating a single element in the latent vector of the lowest frequency band results in a change in a single high-level attribute of images. While similar results have been shown in disentanglement literature like Higgins et al. (2016), our model boasts superior synthesis quality thanks to the generative capabilities of diffusion models.

5. Conclusion

In this paper, we proposed a method of dividing latent variables of diffusion models into multiple groups and assigning interpretable meaning to each group. When extended to the frequency domain, our model can obtain a representation of an image with a multi-level hierarchy. We showed several utilities of such representations including disentanglement and image editing.

References

Albergo, M. S. and Vanden-Eijnden, E. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Albergo, M. S., Boffi, N. M., and Vanden-Eijnden, E. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Chen, R. T., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Choi, J., Lee, J., Shin, C., Kim, S., Kim, H., and Yoon, S. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11472–11481, 2022.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

Fischer, A. and Igel, C. An introduction to restricted boltzmann machines. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings 17*, pp. 14–36. Springer, 2012.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Kim, H. and Mnih, A. Disentangling by factorising. In *International Conference on Machine Learning*, pp. 2649–2658. PMLR, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lee, S., Chung, H., Kim, J., and Ye, J. C. Progressive deblurring of diffusion models for coarse-to-fine image synthesis. *arXiv preprint arXiv:2207.11192*, 2022.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Liu, Q. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022.

Liu, X., Gong, C., and Liu, Q. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Murphy, K. P. *Probabilistic machine learning: Advanced topics*. MIT Press, 2023.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Su, X., Song, J., Meng, C., and Ermon, S. Dual diffusion implicit bridges for image-to-image translation. In *The Eleventh International Conference on Learning Representations*, 2022.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Yeats, E., Liu, F., Womble, D., and Li, H. Nashae: Disentangling representations through adversarial covariance minimization. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pp. 36–51. Springer, 2022.

Zhao, S., Song, J., and Ermon, S. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.

A. Additional Background on Deep Latent Variable Models

One goal of latent variable models (Fischer & Igel, 2012; Goodfellow et al., 2014; Kingma & Welling, 2013) is discovering underlying explanatory factors of data without human supervision. To find a model that explains observations well, one defines $p_\theta(\boldsymbol{x}, \boldsymbol{z}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})d\boldsymbol{z}$ and optimizes

$$\min_{\boldsymbol{\theta}} d(p(\boldsymbol{x})||p_{\boldsymbol{\theta}}(\boldsymbol{x})) \tag{10}$$

for some divergence $d(\cdot)$. Note that $p(\boldsymbol{z})$ is often defined as factorized distribution such as standard Gaussian distribution since we want to figure out independent generative factors. While simply matching marginal distributions does not guarantee that resulting $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is close to true generative process $p(\boldsymbol{x}|\boldsymbol{z})$, learned $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ captures important underlying factors of data with proper inductive biases. While there has been a line of research that utilizes latent variable models for representation learning (Higgins et al., 2016; Kim & Mnih, 2018; Chen et al., 2018), their performance is nowhere close to current self-supervised methods (Grill et al., 2020; Chen et al., 2020; Caron et al., 2021; He et al., 2022; Oquab et al., 2023) in downstream tasks like image classification. One potential reason might be that the generative training objective is overly difficult (Murphy, 2023). To generate realistic images, every pixel-level detail should be modeled, most of which are unlikely to be useful for the downstream tasks we are typically interested in. Until recently, latent variable models have not been so successful in generating complex datasets such as ImageNet, which is a widely used benchmark dataset for downstream evaluation.

B. Related Work and Motivation

Most of the current state-of-the-art disentanglement methods are based on deep latent variable models, especially VAEs (Kim & Mnih, 2018; Higgins et al., 2016; Chen et al., 2018). Higgins et al. (2016) showed that increasing the relative importance of KL term in ELBO encourages disentanglement. This is because the aggregate posterior gets closer to the factorized Gaussian prior (Kim & Mnih, 2018). Yet, the approximate posterior has to be chosen within the limited family of tractable distributions, so it is hard to make aggregate posterior close to prior while maintaining high mutual information between data and latent vector, which causes blurriness (Zhao et al., 2017).

Several follow-up works (Chen et al., 2018; Kim & Mnih, 2018) focused on the fact that disentanglement can be achieved by regularizing the TC penalty term only. Since these methods do not necessarily put more emphasis on forcing the aggregate posterior to be close to the factorized Gaussian prior, their latent variables tend to be more discriminative, so the sample qualities are better than β-VAEs. However, since TC is generally intractable, it needs to be approximated via minibatch statistics (Chen et al., 2018) or mini-max optimization (Kim & Mnih, 2018; Yeats et al., 2022).

Motivation Besides their superior sample qualities, diffusion models carry several attractive properties to help circumvent the aforementioned drawbacks of VAEs. In contrast to VAEs, it is guaranteed in diffusion models that the aggregate inference distribution matches the prior in the optima (see Remark. 2.1), without making a trade-off with sample quality. Additionally, the mutual information between data and learned representation is infinite as the learned mapping from data to noise is bijective. This is desirable since we want to *"disentangle as many factors as possible, discarding as little information about the data."* (Bengio et al., 2013). Finally, there is no need to train additional inference networks as in VAEs.

C. Additional Experiments

C.1. Image editing

Since the inference of diffusion models is invertible, we can apply our method to editing real images. Fig. 8 illustrates that we can control a single attribute of input images by manipulating one element of the obtained representation. Specifically, we first obtain a feature of an image by solving the forward ODE, edit one element of the feature, and reconstruct the image by solving the reverse ODE. Moreover, thanks to a hierarchical structure of latent variables, we can generate diverse variations of an input image, as shown in Fig. 9.

C.2. Optimal time interval for synthesis quality

Recall that we set $\{(t_{\text{start}_j}, t_{\text{end}_j})\}_{j=1}^k$ such that they do not overlap with each other. As shown in Tab. 1, the best result is obtained when the high-frequencies are emphasized in training time and the low frequencies are emphasized in generation

Figure 8: Real image editing results. We manipulate only a single element of the coarsest latent group.

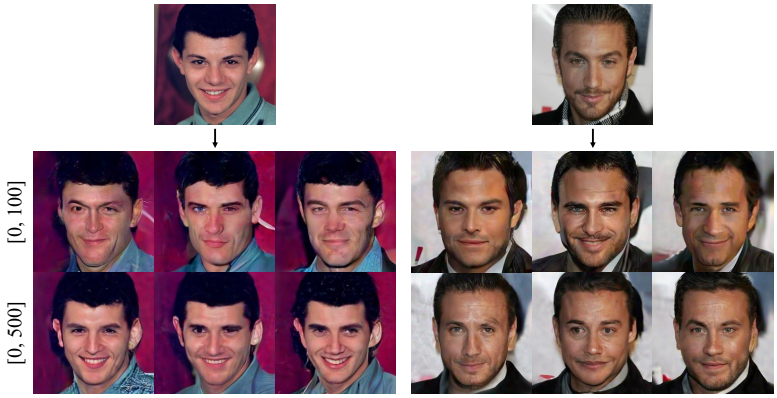


Figure 9: Image variation results. The numbers in parentheses denote the indices of the latent elements shared within the generated images at the same rows.

Table 1: FID10K results for each time interval on FFHQ 64 $\times$ 64 dataset. Note that the time intervals need to be chosen separately for training and generation time. For instance, the top left cells indicate the FID result when t_{end_1}, t_{end_2}, and t_{end_3} are $0.1, 0.2, 0.5$ and $0.1, 0.8, 0.9$, respectively.

Training / Generation	0.1, 0.8, 0.9	0.3, 0.8, 0.9	0.7, 0.8, 0.9	0.1, 0.2, 0.9	0.1, 0.5, 0.9	0.1, 0.2, 0.3	0.1, 0.2, 0.5
0.1, 0.2, 0.5	25.39	29.47	34.77	25.58	24.97	25.98	25.74
0.3, 0.4, 0.7	21.38	29.39	34.54	21.16	20.63	21.42	21.23
0.5, 0.6, 0.9	21.48	31.68	36.16	20.74	20.40	20.66	20.62
0.6, 0.8, 0.9	19.69	27.75	32.83	17.60	17.76	**17.41**	17.55
0.7, 0.8, 0.9	22.59	34.21	39.61	20.72	20.83	20.31	20.42

time. This is an interesting observation that contrasts with previous work, where the best results are obtained when coarse aspects are given more weight during training (Ho et al., 2020; Choi et al., 2022), and fine details are emphasized in sampling time (Dhariwal & Nichol, 2021).

D. Implementation Details

Throughout our experiments, we use DDPM++ architecture (Song et al., 2020) from the code of Karras et al. (2022)[1]. We use their script for computing FID, and most of the training and model configurations are also adapted from Karras et al. (2022). The random seed is fixed to 0 in all experiments. We linearly anneal the learning rate as in previous work (Karras et al., 2022; Song et al., 2020).

[1] https://github.com/NVlabs/edm