Electronic Supplementary Material (ESI) for Soft Matter.

Electronic Supplementary Information (ESI) for **MD-GAN with multi-particle input: the machine learning of long-time molecular behavior from short-time MD data**

Ryo Kawada,[1] Katsuhiro Endo,[1] Daisuke Yuhara,[2] Kenji Yasuoka[1*]

[1]Department of Mechanical Engineering, Keio University,
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, 223-8522, Japan

[2]Materials Design Laboratory, Science & Innovation Center,
R&D Transformation Div.,Mitsubishi Chemical Holdings Group,
1000 Kamoshida-cho, Aoba-ku, Yokohama, Kanagawa, 227-8502, Japan

[*] E-mail:yasuoka@mech.keio.ac.jp (K.Y.)

This PDF file includes:

Section 1. Effects of introducing latent variables and distribution stabilization mechanisms

Section 2. Detailed architecture of MD-GAN

Section 3. How to calculate err

Supplementary Fig. S1 Schematic view of time evolution in a low-dimensional space

Supplementary Fig. S2 Detailed architecture of MD-GAN

Section 1. Effects of introducing latent variables and distribution stabilization mechanisms

MD-GAN succeeds in reducing the exposure bias by introducing latent variables and distribution stabilization mechanisms. In general, most real-world data can be considered as lying along a low-dimensional manifold in a high-dimensional space[1, 2, 3]. Based on this idea, the evolution of the extracted trajectory of the subsystem is also represented as the evolution of dense latent variables, embedded in a low-dimensional space. Supplementary Fig. 1 shows the effects of these latent variables. The left figure is a conceptual figure of time evolution of a raw trajectory in high-dimensional space wherein no latent variables are introduced, while the right figure shows a conceptual figure of time evolution in low-dimensional space wherein latent variables are introduced. Each sphere represents M steps of trajectory information. In the figure on the left, the gray sphere is $\boldsymbol{Y}_{k*M:(k+1)*M-1}$, and the red and yellow spheres represent examples of points that might be taken as $\boldsymbol{Y}_{(k+1)*M:(k+2)*M-1}$ after the next time evolution. In the figure on the right, the gray sphere is $\boldsymbol{z}_i$, and the red sphere and yellow sphere represent examples of points that may be taken as $\boldsymbol{z}_{i+1}$ next. For visualization purposes, the figure on the left is drawn in three dimensions, but the actual dimension is $3M$ for single-particle input. The reason for the $3M$ dimension is that $\boldsymbol{Y}_{k*M:(k+1)*M-1}$ is the information in xyz coordinates for M steps. Similarly, for the figure on the right, the latent variable may be more than two-dimensional(4 dimensions for single-particle input). The blue region represents the support where variables $\boldsymbol{z}_i$ and $\boldsymbol{Y}_{k*M:(k+1)*M-1}$ can exist respectively, and it is called the manifold. When the gray sphere has information of the current M steps, it can take various states via stochastic transition, during the next time evolution. In this evolution, although the red sphere is on the manifold, the yellow sphere is off it. Assuming that a yellow sphere is input to MD-GAN in the estimation phase, the input is an unknown input in the training phase, and this may result in exposure bias. In the time evolution of a high-dimensional and sparse space, as shown in the left figure, it is likely to deviate from the manifold, while in a low-dimensional and dense potential space, as shown in the right figure, it is less likely to deviate from the manifold. Therefore, the introduction of latent variables is expected to make it harder to deviate from the manifold through time evolution and contribute to the reduction of exposure bias.

The initial latent variable $\boldsymbol{z}_0$ is sampled from the n_z-dimensional uniform distribution $U(S^{n_z})$. G_z generates further $\boldsymbol{z}_i$ using the previous $\boldsymbol{z}_{i-1}$ and a random value. We then apply G_z to $\boldsymbol{z}_0$ and repeatedly generate $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, and so forth. After $\boldsymbol{z}_0$ is time evolved by G_z, there is no guarantee that the distribution of $\boldsymbol{z}_1$ will be as uniform as that of $\boldsymbol{z}_0$. If they are different, the trajectories generated by G_Y from $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$ will not satisfy the third assumption of MD-GAN. If the trajectories generated from $\boldsymbol{z}_0$ and $\boldsymbol{z}_1$ are used for training, then $\boldsymbol{z}_1$ is input to G_z in estimation phase, and this implies an unknown input and leads to the occurrence of exposure bias. Therefore, G_z is applied repeatedly until the distribution of latent variables becomes stationary. After seven or more time evolutions, the latent variables obtain a stationary distribution and trajectories are generated from these latent variables. Thus, the mechanism that evolves in time until a stationary distribution is obtained is the distribution stabilization mechanism, which contributes to the reduction of exposure bias.

Section 2. Detailed architecture of MD-GAN

Supplementary Fig. 2 shows the overall architecture of MD-GAN, and the architectures of G_Y, G_z, and D_G are shown in Supplementary Fig. 2A, B, and C, respectively. Supplementary Fig. 2 describes the parameters used for the single-particle input. For the three-particle input, the number of channels for convolution was increased from three to nine, and the shape of y_k was set to 64×9. The optimizer was Adam[4], and the batch size was 64. The architecture of MD-GAN in a previous study[5] was based on U-Net. However, for ease of use, we changed the architecture to consist mainly of affine layers. There was no significant change in the execution results owing to architectural changes. In addition, the previous study used two discriminators: D_G and a discriminator that calculates the Wasserstein distance of two latent variables before and after the time evolution. In this study, we did not use the discriminator between the two latent variables.

Section 3. How to calculate err

In this paper, err, the evaluation index, is defined as

$$err := \int_{s_1}^{s_2} f(t)ds / \int_{s_1}^{s_2} ds \tag{1}$$

In the MSD prediction, $s = \log_{10} t$, s_1 corresponds to the time when $n = 1/2$ diffusion ends, transition to $n = 1$ diffusion begins, and s_2 corresponds to the time when the transition ends and the normal diffusion to $n = 1$ begins. When calculating err, appropriate discretization should be performed depending on the number of step skips. Transforming Eq. 1 into an expression with respect to t:

$$\int_{s_1}^{s_2} f(t)ds / \int_{s_1}^{s_2} ds = \int_{t_1}^{t_2} f(t)\frac{1}{t}dt / \int_{t_1}^{t_2} \frac{1}{t}dt \tag{2}$$

Let $t_k = k\Delta t$ be the width of the time increment to be discretized, where Δt is the tick width after applying step skip, and is thus the product of the number of step skips and the output tick width of the MD data. k is the number of steps after applying step skip. Discretizing Eq. 2, we get

$$\sum_{k \in transition\, area} f(k\Delta t)\frac{1}{k\Delta t}\Delta t / \sum_{k \in transition\, area} \frac{1}{k\Delta t}\Delta t \tag{3}$$

$$= \sum_{k \in transition\, area} f(k\Delta t)\frac{1}{k} / \sum_{k \in transition\, area} \frac{1}{k} \tag{4}$$

From the above equation, the value of err in the MSD prediction can be obtained. In the prediction of the end-to-end vectors, considering $s = t$, we get

$$\sum_{k \in area\, in\, which\, C(t)\, is\, greater\, than\, 0.1} f(k\Delta t)\Delta t / (t_2 - t_1) \tag{5}$$

References

[1] Basri, R., Jacobs, D.W.: Lambertian reflectance and linear subspaces. IEEE transactions on pattern analysis and machine intelligence **25**(2), 218–233 (2003)

[2] Basri, R., Jacobs, D.: Efficient representation of low-dimensional manifolds using deep networks. arXiv preprint arXiv:1602.04723 (2016) https://arxiv.org/abs/1602.04723

[3] Lee, K.-C., Ho, J., Yang, M.-H., Kriegman, D.: Video-based face recognition using probabilistic appearance manifolds. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 1, p. . IEEE, ??? (2003)

[4] Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs] (2017) https://arxiv.org/abs/1412.6980

[5] Endo, K., Tomobe, K., Yasuoka, K.: Multi-Step Time Series Generator for Molecular Dynamics. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (2018)

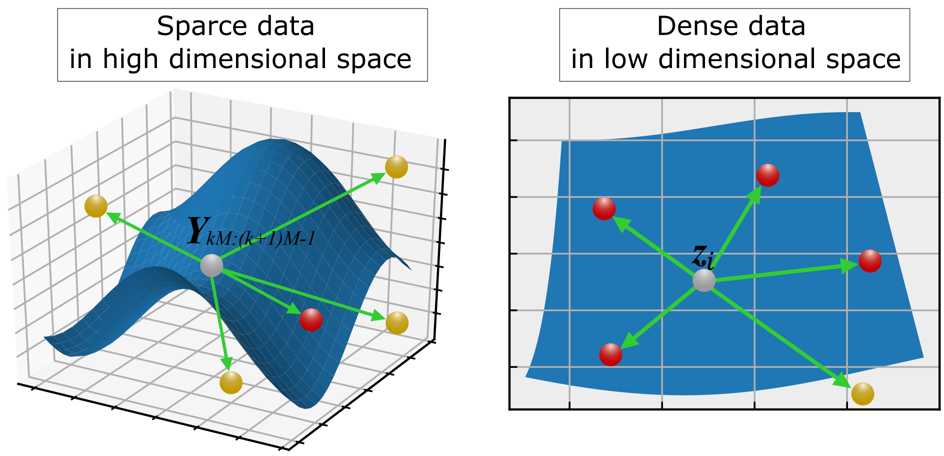

Fig. S 1. Schematic view of time evolution in a low-dimensional space. The gray, red, and yellow spheres contain the trajectory information for M steps of the extracted subsystem. The blue region is the manifold where the information of M steps can be found.

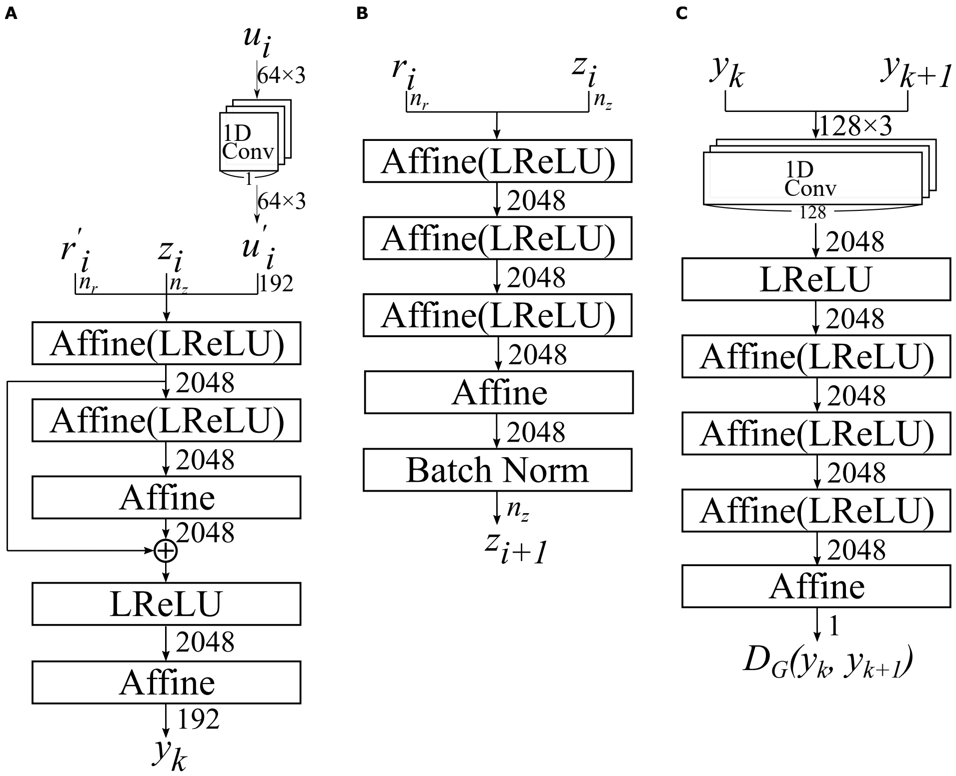

Fig. S 2. Detailed architecture of MD-GAN. r_i, r'_i are random numbers sampled from the uniform distribution, u_i are random numbers sampled from the normal distribution, z_i is a latent variable, n_r is the dimension of the random number, n_z is the dimension of the latent variable, and y_k is the trajectory for 64 steps. The numbers next to the arrows represent the shape of the output (batch size is omitted). **(A)** Architecture of G_Y In one-dimensional convolution, the kernel size is 1, the stride is 1, and the number of input and output channels are both 3. **(B)** Architecture of G_z **(C)** Architecture of D_G In one-dimensional convolution, the kernel size is 128, the stride is 128, the number of input channels is 3, and the number of output channels is 2048.