



Title	Time-series prediction of velocity trajectories using generative AI for estimating molecular properties
Author(s)	Tanaka, Shota; Shibahara, Masahiko
Citation	Molecular Simulation. 2025, 51(11), p. 768-779
Version Type	AM
URL	https://hdl.handle.net/11094/104467
rights	This article is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Time-series prediction of velocity trajectories using generative AI for estimating molecular properties

Shota Tanaka¹ and Masahiko Shibahara^{1*}

¹ Graduate School of Engineering, Osaka University
2-1 Yamadaoka, Suita 565-0871, Osaka, Japan

* Corresponding Author: siba@mech.eng.osaka-u.ac.jp

Keywords: Generative AI, Molecular dynamics, Neural networks, Molecular properties

Abstract

As the high computational cost of molecular dynamics (MD) simulations is a significant issue, it is important to investigate the applicability and performance of Generative Adversarial Networks (GANs), known as one of the machine learning approaches. In the present study, molecular information for liquid argon (Ar) systems at various temperatures and densities in equilibrium states was employed as an example of prediction via GANs. Time series data of molecular velocities calculated using MD were extracted as training samples for the GANs, and the predictive capability of the GANs was evaluated by examining the reproducibility of sample distributions and self-diffusion coefficient from different calculation methods. It was demonstrated that the samples of molecular velocity generated by the GANs replicated the MD simulation distributions. Mean squared displacements (MSD) and velocity autocorrelation functions (VACF) calculated from the generated velocity trajectories via GANs showed qualitative reproducibility of those calculated by the MD simulations. Regarding the self-diffusion coefficient, we found that the GANs reproducibility of the MD results tends to decline under conditions of low density and high temperature. Furthermore, it was found that the self-diffusion coefficients predicted using GANs did not significantly depend on the calculation methods used for the self-diffusion coefficient.

1. Introduction

Molecular dynamics (MD) simulations are frequently used to investigate dynamical phenomena and structural properties at the nanoscale. Its versatility is exemplified by its applications in various situations such as ice nucleation of water in the bulk system [1], heat transfer in the nanochannels of Cu [2], diffusion phenomena and viscosity of water [3] and so on. On the other hand, the complexity and the scale of recent simulation targets have increased, resulting in a notable increase in calculation costs. To address this issue, machine learning-assisted approaches have been recognized as promising solutions. For example, studies of MD simulations using the Machine Learning potential (ML potential) for interatomic potentials were reported as efficient methods to reduce the high-load processes of interaction calculations with density functional theory [4, 5]. Recent machine learning based methods for analyzing and enhancing MD simulations were discussed to obtain equilibrium properties, such as the free energy [6] and the ML based ensemble learning strategies were employed to predict the partition function of molecular fluids absorbed in metal- and covalent organic frameworks [7]. The integration of ML techniques into the multiscale molecular simulation methods was introduced for bulk complex chemical systems like polymers [8].

On the other hand, some authors have suggested the application of machine learning prediction systems to material informatics. In those cases, the neural network (NN) model is expected to acquire the relationship between structural patterns and physical properties by simulating only limited structural patterns. While the computational cost of MD simulations makes it impractical to explore extensive design spaces, the use of NN models allows for rapid screening of the entire design space within just a few seconds [9, 10]. In this work, around 100 sets of training data were typically prepared as the initial training dataset, but their limited accuracy necessitated an iterative process. By alternating between screening and MD simulations, the number of training datasets was gradually increased, finally resulting in an NN model ready for practical use. The authors found desirable candidates out of numerous patterns in the design space using such an optimized NN model. In addition to the use of NN, a genetic algorithm has also been employed for the optimization of superlattice and random multilayers created from diamond structured Si and Ge with respect to thermal transport [11].

Furthermore, a Recurrent Neural Network (RNN), which is well-known in the field of Large Language Models, was applied for predicting time-series data. As example of that work, Wang et al. obtained the temporal evolution of molecular velocities and positions of Si atoms from ab initio computations for the RNN learning, which was used for multi-step future prediction, and they reported successful prediction [12]. The examples shown above have demonstrated the potential of hybrid approaches combining machine learning and MD simulations to reduce computational costs.

Generative AI has started to be utilized in the field of MD simulations. Generative Adversarial Networks (GANs) have already been applied in the field of drug discovery [13, 14], and more recently, applications have begun to appear in the field of time series prediction of molecular properties. For example, the combination of variational autoencoder (VAE) and RNN were applied to the polycrystalline nickel system [15]. In that study, the authors extracted the temporal 2D image data of atomic configuration from MD simulations, and they convoluted the data into latent variables with the encoder. The RNN model was then trained to acquire the time-development of the latent variables, and the decoder reconstructed the time-developed data from the latent variables predicted. Furthermore, GANs are being applied to the prediction of displacement trajectories in polymer systems [16, 17]. The GANs acquired the stochastic time evolution of molecules instead of the deterministic evolution of MD through short-term trajectories and are able

to predict long-term trajectories. In other work, GANs are being applied to the prediction of heat flux trajectories in metallic systems; they predicted the thermal conductivity of Al and Cu through the use of the Green–Kubo relation [18]. These studies show the possibility of applying generative AI applicability for the prediction of molecular properties in MD simulations via the time-series prediction of physical quantities. On the other hand, due to the limited number of relevant studies, it remains unclear how the characteristics of the sample data influence the predictions of the generative AI in the literature. Moreover, generative AI has predominantly been applied to complex systems, such as polymer systems, while its application to simpler cases, such as monoatomic systems, has not yet been adequately reported. Reports of the application of generative AI in simple molecular systems will facilitate a deeper understanding of the relationship between the molecular properties obtained by the molecular dynamics simulations and the prediction performance of the generative AI. Furthermore, in the field of ML potentials, unique architectures, such as the well-known SchNet [19] model, have been developed, which are widely adopted as baseline models due to their effectiveness. In contrast, the generative AI methods are still in development, and no standard or baseline models currently exist in their application in MD simulations. As a result, designing prediction models of generative AI for the prediction of molecular properties is highly challenging, and the methodologies for improving them remain insufficiently developed.

In the present study, we adopted a GANs prediction method similar to those used in previous studies [17], with modifications to some hyperparameters and model architectures. The developed method was applied to predict the time-series of molecular velocities in liquid argon (Ar) at a thermal equilibrium state and its prediction ability tested by comparison to MD results. It is well known that the self-diffusion coefficient is strongly dependent on the density and temperature of the liquid [20, 21]. To evaluate the effects of density and temperature on the self-diffusion coefficient, the time-series data of molecular velocities were sampled from MD simulations while varying the temperature and density of the molecular system. In previous studies, molecular displacement trajectories were used as training samples, achieving accurate predictions of the self-diffusion coefficient via MD-GAN [16, 17, 22]. In the present study, we employed molecular velocity trajectories as training samples and evaluated their performance in predicting the self-diffusion coefficient. Furthermore, the self-diffusion coefficient can be calculated using two distinct methods: usage of the Green–Kubo relation, which directly calculates the self-diffusion coefficient from the velocity autocorrelation function (VACF), and usage of the Einstein relation, which employs the MSD. We investigated the prediction accuracy for the two types of self-diffusion calculations and how the density and temperature of the molecular system influence both the velocity trajectory data and the predictive performance of the GANs.

2. Methodology

2-1. GANs System

The prediction system of the GANs used in the present study is based on the system called MD-GAN in the previous work [16, 17, 22]. The training flow of the architectures of the GANs is shown in Fig. 1, and the detailed architectures are shown in the Supplementary Information (SI). In the MD simulation, the positions and velocities of all molecules are updated by calculating forces in short time increments following Newton’s equation. The time-series data are determined by the initial settings. On the other hand, in the MD-GAN, only the trajectories of their subsystems to obtain the desired physical quantity are necessary, and their stochastic time evolutions become long trajectories of the subsystems. In the multi-step prediction using one generator recursively, the normal GANs generator G_X is expected to acquire the time evolution of the subsystems extracted as training samples. Put in detail, the time-series data of length l at the previous state $X_{i:l:(i+1)*l}$ is taken as the input of G_X , and G_X outputs the probabilistically possible trajectories in the next step $X_{(i+1)*l:(i+2)*l}$ from the previous state. GANs generator can acquire the time evolution supposing a Markov model in the multi-step prediction system of the normal GANs. We can express the sample of the stochastic time evolution as a conditional probability as follows:

$$X_{j:l:(j+1)*l} \sim p(X_{j:l:(j+1)*l} | X_{(j-1)*l:j*l}) \quad (1)$$

Here, by assuming that the next-step data depends only on the previous data (Markov property) and that the stochastic time evolution is stationary, it becomes possible to predict the next-step data. However, the exposure bias which occurs inevitably becomes a problem in such a system. To mitigate the exposure bias, MD-GAN utilizes an additional generator G_h for latent variables h in addition to the generator, the discriminator used in the typical GANs. Instead of the time evolution of the subsystems X , the latent variables h embedded into a low-dimensional information space are employed. Then, G_h is expected to acquire the time evolution tendency of the latent variables through sampling:

$$h_{j+1} \sim p(h_{j+1} | h_j) \quad (2)$$

The generator reconstructs the time-series data from this latent variable h as the following equation:

$$X_{j:l:(j+1)*l} \sim p(X_{j:l:(j+1)*l} | h_j) \quad (3)$$

It becomes possible to address the issue of exposure bias by considering such an indirect generation approach supposing a hidden Markov model. According to previous work [17] that discussed and examined n_h , when the dimensionality n_h of the latent variables is too low, the accuracy of prediction decreases. Therefore, we use n_h set to 32 which is checked following the previous work [17]. The initial distribution u for the latent variables is sampled from a uniform distribution in the range $[0,1]$. By performing repeated generation with G_h for a random number of times between 12 and 15 times from the initial distribution, the distribution relaxation is achieved, and we call this process the distribution relaxation phase. The gradient updating in the training phase and data sampling in the prediction phase are initiated after the distribution relaxation phase. In the present study, the temporal data length l for one output of the GANs was set to 64 based on the value employed in the previous study, of which the influence was

discussed in section 4 of the supplemental information. As the loss function in the GAN, we adopted the Wasserstein Loss and zero-centered Gradient Penalty referring to references [23, 24] and fixed the regularization due to better convergence of training as shown in SI (see Fig. S 1). Consequently, the loss function is defined as follows:

$$L = E[D_X(X)] - E[D_X(\tilde{X})] + \frac{\lambda}{2} E \left[(\| \nabla D_X(\hat{X}) \|_2)^2 \right] \quad (4)$$

where, λ is the gradient penalty coefficient set to 20, $\hat{X}$ is the value interpolated between the generated data X and training data $\tilde{X}$. During the training phase, for each gradient update, the generator and the latent variable generator performed three time-evolution steps to produce $X_{j*l:(j+3)*l}$, generating the total data length of 192 as three segments, each with a length of 64. The discriminator was used to obtain the Wasserstein Loss between the training sample $\tilde{X}_{j*l:(j+3)*l}$ and the generated sample $X_{j*l:(j+3)*l}$. The generator and the latent variable generator were updated to minimize the loss, while the discriminator was updated to maximize the loss. In the WGAN-GP, it is recommended that the discriminator should update more often than the generators. Hence, for each update of the gradient of the generator and the latent variable generator, the gradient of the discriminator was updated five times. The learning rates of the generator and the latent variable generator were set to 2×10^{-6} and that of the discriminator was set to 4×10^{-7} . RMSprop was adopted as the optimization method, and the batch number of training phases was set to 64 and that of prediction phases was set to 32. In the present study, we sampled the average of the molecular velocities in several steps from the MD simulations. When we average molecular velocities of MD trajectories in 15 steps, Δt_{GAN} is equivalent to 30 fs whereas Δt_{MD} is 2 fs. We define these sampling steps as ‘averaging steps’, and the value was set to 15 in the present study. The training data is normalized by the standardization as a pre-processing step. This is because we have confirmed that the predicted self-diffusion coefficient becomes almost constant when we change the averaging steps from 8 to 16 as shown in section 5 of the supplemental information and we need to reduce the number of MD data required due to the excessively large averaging steps. We prepared two datasets, one for learning and one for testing, and we used the first 0.9 ns of the 3.0 ns test data as training data. Extra data for validation is generally not required in unsupervised learning like GANs. The gradients were updated for a total of 900000 iterations in the training phase, and the long velocity trajectories in the prediction phase were generated every 5000 times during the last 450000 iterations. Each prediction phase generated 2880 long velocity trajectories of 100000 data-length. The learning model and system were built on TensorFlow, a framework that can be run in Python. We employed the environment of Python 3.11.5, TensorFlow 2.12.0, cuDNN 8.2.1 and CUDA v12.2 or 12.4 in the present study.

In machine learning studies, the method of domain-knowledge is often introduced to improve prediction accuracy. For example, in the previous study [22], the molecular motion is assumed to be centered on 0 if the number of samples is large enough. In the present study, the assumptions of the isotropy of molecular motions and stationary center of mass after the long-time average were given, and the following corrections could be made to the velocity trajectories in the prediction phase.

$$v'_{i,\alpha} = \left(v_{i,\alpha} - \frac{1}{N} \sum_{i=1}^N v_{i,\alpha} \right) / s_\alpha \quad (4)$$

where, α is the element of the cartesian coordinates of (x, y, z) .

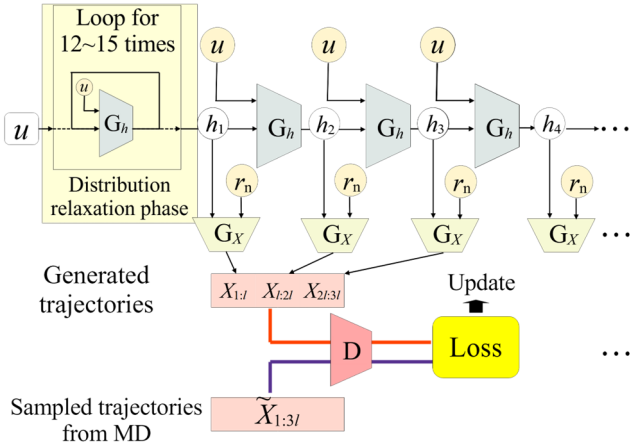


Fig. 1 Prediction flow of the GAN system in the training phase employed in the present study; r_n is the random number in the standard normal distribution, u is the random number of the uniform random distribution ranged in $[0,1]$, G_h is the latent variables generator, G_X is the generator, D is the discriminator and h_j is the latent variable of the j -th prediction

2-2. MD simulation

We prepared the calculation conditions of density and temperature from case 1 to 15 as shown in Table 1. The system at each

calculation condition can be confirmed as liquid from the phase diagram including the extreme conditions [25]. Figure 2 shows a snapshot of the calculation system in case 3. The unit cell of each condition and the number of argon molecules were set as follows:

- case 1-case 3: $(x, y, z) = (43.6, 43.6, 43.6) \text{ \AA}$, 1372 molecules
- case 4-case 6: $(x, y, z) = (42.3, 42.3, 42.3) \text{ \AA}$, 1372 molecules
- case 7-case 9: $(x, y, z) = (47.2, 47.2, 47.2) \text{ \AA}$, 2048 molecules
- case 10-case 12: $(x, y, z) = (51.7, 51.7, 51.7) \text{ \AA}$, 2916 molecules
- case 13-case 15: $(x, y, z) = (56.1, 56.1, 56.1) \text{ \AA}$, 4000 molecules

Periodic boundary conditions were employed in all directions. We adopted liquid Ar molecules as a simple monoatomic molecule, and the intermolecular potential was described using the 12-6 Lennard-Jones potential shown below.

$$U(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right] \quad (5)$$

where, ϵ/k_B and σ are 120 K and $3.405 \times 10^{-10} \text{ m}$, respectively, as in a previous study [21]. The time step was set to 2.0 fs and the integration of Newton's equation was performed using the Leap-Frog algorithm. Temperature was initially controlled to a target value by the velocity scaling method in 6 ns. The calculation systems were relaxed as an NVE ensemble for 10 ns after velocity scaling, followed by sampling of physical quantities. The time-series data of molecular velocity was obtained during 450000 MD steps. In the present study, the average steps were set to 15 as described in Section 2.1; therefore, we regarded Δt_{MD} as 2.0 fs and Δt_{GAN} as 30.0 fs. MD simulation was conducted with Δt_{MD} for 1500000 steps (3 ns), and then the data during the first 450000 steps (0.9 ns) in the sampling phase was used as training samples of an equivalent of 100,000 steps for the GANs.

Table 1. Density and temperature conditions in MD simulations.

		Temperature		
		100 K	120 K	140 K
Density	1098 kg/m ³	case 1	case 2	case 3
	1200 kg/m ³	case 4	case 5	case 6
	1291 kg/m ³	case 7	case 8	case 9
	1400 kg/m ³	case 10	case 11	case 12
	1501 kg/m ³	case 13	case 14	case 15

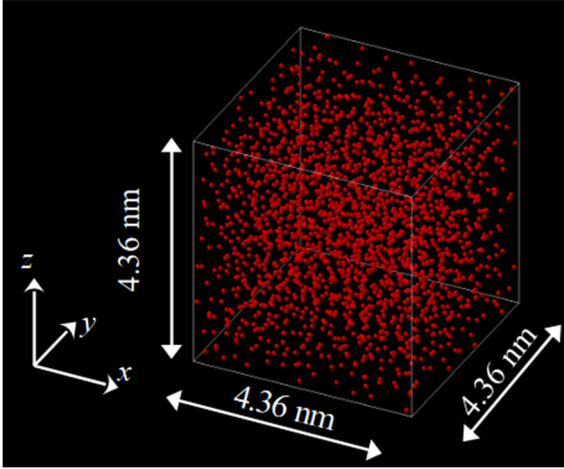


Fig. 2 A snapshot of calculation system (case 8)

2-3. Calculation of physical properties

We evaluated the performance of GANs by calculating the following properties.

- Velocity distribution of molecules
- Diffusion properties calculated via the Einstein relation [26]
- Diffusion properties calculated via the Green-Kubo relation [26]

We assessed the velocity distribution of the entire set of predicted values comparing with that of MD samples at first. In equilibrium systems, the probability density of velocity of molecules follows the Maxwell distribution:

$$f(v) = 4\pi v^2 \left(\frac{m}{2\pi k_B T} \right)^{\frac{3}{2}} e^{-\frac{mv^2}{2k_B T}} \quad (6)$$

When we calculate the self-diffusion coefficient using the Einstein relation, we have to obtain the displacement trajectories by integrating the velocity trajectories with the leapfrog method and calculate the MSD as follows.

$$\text{MSD}_\alpha(t) = \langle [x_{i,\alpha}(t) - x_{i,\alpha}(0)]^2 \rangle \quad (7)$$

We can calculate the gradient of the fitting line between time t and MSD by using the least square method. Assuming an isotropic

and unbiased system, we can consider the average of the self-diffusion coefficients for each directional component. Therefore, the self-diffusion coefficient can be calculated as follows according to the Einstein relation:

$$D_1 = \lim_{t \rightarrow \infty} \frac{d}{dt} \left(\frac{\text{MSD}(t)}{6} \right) \quad (8)$$

We refer to this method as “method 1” in the subsequent section.

In addition, the self-diffusion coefficient can be calculated by integrating the VACF through Green–Kubo relation.

$$D_2 = \frac{1}{3} \int_0^\infty \langle \mathbf{v}_i(t) \cdot \mathbf{v}_i(0) \rangle dt \quad (9)$$

Here, VACF is expressed as follows.

$$\text{VACF} = \mathbf{v}_i(t) \cdot \mathbf{v}_i(0) \quad (10)$$

When calculating the self-diffusion coefficient, we employ the ensemble average of the velocity autocorrelation function over the number of molecules in the system. This calculation method is widely recognized as the Green–Kubo relation, and we refer to this method as “method 2” in the later section.

The ensembles of MSD and VACF were sampled at 50 ps intervals of both the generated trajectories via GANs and those found by MD. The self-diffusion coefficient at the condition of 1290 kg/m³ and 100 K was calculated using both methods of Einstein relation and Green–Kubo relation. These values are $D_1 = 3.525$ and $D_2 = 3.511$ m²/s, respectively, and since the results were confirmed to be consistent with the previous study [21], we consider the computational method employed in the present study to be valid.

3. Results and discussions

We evaluated the performance of the GANs prediction for molecular properties with a seed value that changed five times randomly, and in MD simulation, we calculated the molecular properties changing the initial position and velocity five times randomly. To avoid showing similar graphs for molecular properties, only one result is shown in all figures, which was obtained from one specific seed. The self-diffusion coefficients were calculated as the average value for these five calculations regardless of MD simulations and GANs predictions. In Section 3, the calculation results are shown only in case 8, which is the center point of all the conditions (case 1-15) and the related results in other cases are shown in section 3 of SI.

3-1. Velocity distribution predicted by GANs

Figure 3 shows the probability density distribution of predicted velocities (orange region), the distribution of MD samples (blue region) and the Maxwell distribution (black line) under the conditions of case 8. The probability density distributions of all cases are shown in Fig. S 3 of SI. The overlapping of blue and yellow regions makes it difficult to distinguish between them because there is only a slight difference between them. We can speculate that the small difference between MD samples and the Maxwell distribution originates in the small number of samples. Under the given conditions, the predicted velocities were distributed over 0-700 m/s, with a peak in the range of 200 to 300 m/s, as indicated by the Maxwell distribution. The distributions of the predicted velocities and the sampled data from MD using averaging steps are well aligned, and they demonstrated that the GAN accurately reproduced the velocity distribution over the entire regime.

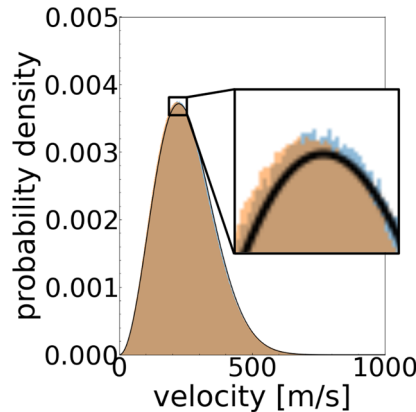


Fig. 3 The probability density distribution of GANs (orange region), MD samples (blue region) and Maxwell distribution (black line)

3-2. Diffusion coefficient calculated via method 1

In method 1, since the output of GANs consists of velocity trajectories, the self-diffusion coefficient cannot be directly obtained from the output as in method 2, making it necessary to calculate the displacements trajectories. Each displacement trajectory via GANs was obtained by means of the integration of the predicted molecular velocity in a similar way to that for the MD simulation.

Figure 4 shows the relationship between t and MSD in case 8. In Fig. 4 (a) and (b), linear and logarithmic scales are used in the vertical and horizontal axes, respectively. If we assume a power-law relation between t and MSD as $\text{MSD} \propto t^n$ where the exponent n might depend on t , then we can calculate n from the gradients of the logarithmic MSD-time plots using the least squares method. The MSDs on a linear scale for all cases are shown in Fig. S 4 of SI. As shown in Fig. 4 (a), the GANs tended to underestimate the MSD calculated by MD. This trend was observed in all cases in Table 1. Figure 4 (b) shows the MSD had similar trends in the range from 1 to 50 ps regardless of whether the GANs or MD were used. Although there is a discrepancy between the MSD calculated by MD and that via GANs up to 1 ps in Fig. 4 (b), the differences up to 1 ps can be ignored for the diffusion coefficient calculations. When we use the Einstein relation, we should confirm its linearity in accordance with $\text{MSD} \propto t^1$. The MSD for MD and GANs had a similar gradient to the $n = 1$ line after 1 ps on a logarithmic scale as shown in Fig. 4 (b). The time transition of the exponent of power fitting t to the function of t^n is determined from Fig. 4 (c), where the points were obtained at every 20 steps. Judging from Figs. 4 (a)-(c), we can safely employ the gradient of MSD in the time range from 10 to 50 ps to calculate the self-diffusion coefficient from MSD in the following section. As the same trends were observed in all cases in Table 1, we also employed MSD during this time period to calculate the self-diffusion coefficients in the other cases.

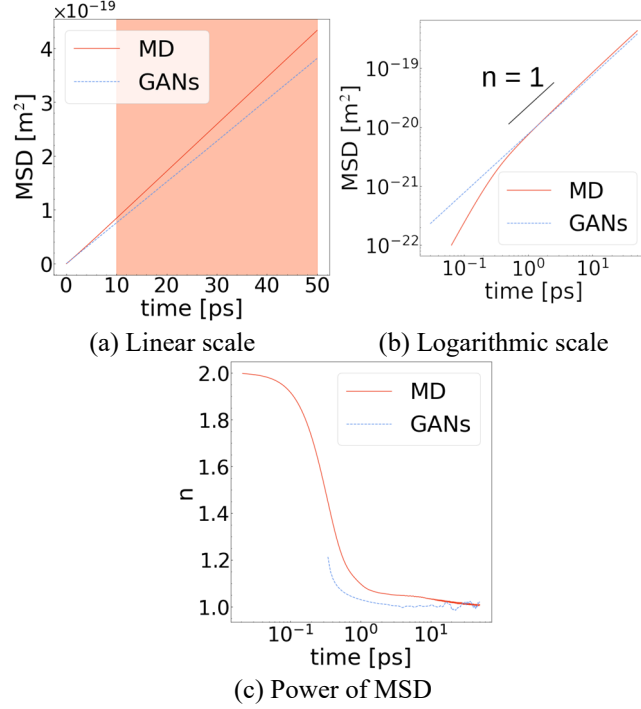


Fig. 4 Relationship between t and MSD: MD simulation (red line) and GANs (blue line)

3-3. Diffusion coefficient calculated via method 2

In this study the GANs output the velocity trajectories, allowing us to calculate the VACF directly and the self-diffusion coefficient through method 2.

Figure 5 shows the relationship between t and VACF, as well as t and D_2 , the self-diffusion coefficient, of the MD simulations and GANs predictions for case 8. The VACF and self-diffusion coefficients of all cases are shown in Fig. S 5 of SI. The VACF for MD converged to zero by 10 ps while, in contrast, that of GANs did not converge but fluctuated at a constant frequency and amplitude. This behavior was observed in all cases in Table 1. Hence, as expected, the value of the self-diffusion coefficient did not converge to a constant value via the prediction of GANs, as shown in Fig. 5 (b), because it was calculated by integrating the VACF. As shown in Fig. 5 (a) and (b), the period of the observed fluctuations is about 2 ps in the GANs prediction and, as in the present study we set the temporal data length of one GANs output to 64 and averaging steps to 15, then Δt_{MD} was set to 2 fs. Thus, the time-length of one GANs output was equivalent to 1.92 ps. That means the GANs in the present study tended to acquire limited patterns of motions and output repeated similar trajectories.

There was a gap between the period of time-length per one GANs output and the ensemble time-length. To confirm the convergence of the VACF, we took the moving average of the VACF of GANs. By using the moving average here, we can obtain a smooth curve, and it converged to zero as shown in Fig. 5 (c) and (d). In contrast to the VACF and D_2 shown in Fig. 5 (a) and (b), the repeated fluctuation does not appear in the MSD results as shown in Fig. 4 (c). Hence, we can confirm that the VACF fluctuates around zero and we derive the self-diffusion coefficient by averaging the range of D_2 from 30 to 50 ps.

As a qualitative trend, the self-diffusion coefficient of GANs was underestimated compared to that of MD as shown in Fig. 5 (d), and this trend also appeared in all cases, similar to that seen in method 1.

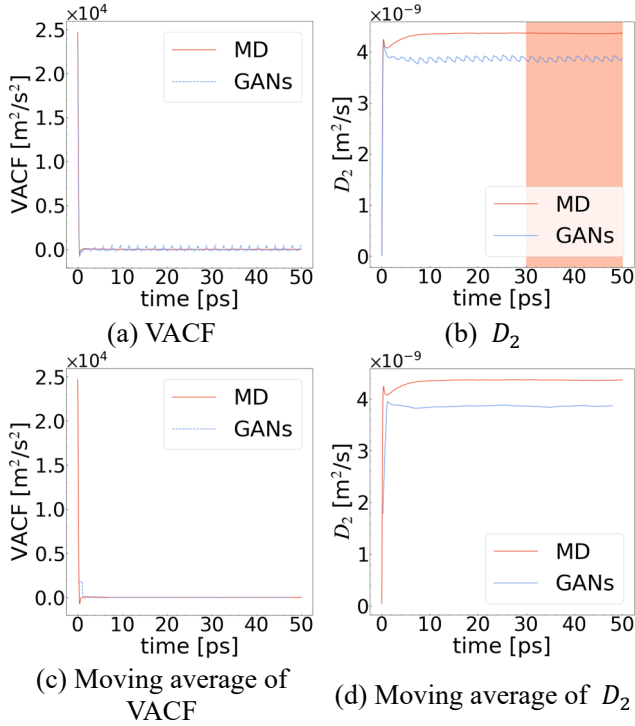


Fig. 5 Relationship between t and VACF or D_2 of MD simulation (red line) and that of GANs (blue line) in case 8. The window size of the moving average is taken as 2 ps in all data

3-4. Comparison of MD simulation and GANs prediction

To evaluate the prediction accuracy of GANs, we calculated the relative error of the self-diffusion coefficient to that calculated by MD. The self-diffusion coefficient was calculated from the MD simulations by changing the initial condition five times for each condition in Table 1. We used the average of the self-diffusion coefficients of these five MD simulations as the true value when we obtain the relative error of the predicted value via GANs. The self-diffusion coefficients in each case are shown in Table 2, and the absolute error value and relative error value are shown in Table 3. Figure 6 shows the relationship between liquid density and self-diffusion coefficient, and reveals that the absolute error of the self-diffusion coefficient decreased as the density increased. Figure 7 shows the relationship between liquid density and relative error of the self-diffusion coefficient in Fig.6, and reveals that the relative error of the self-diffusion coefficient tended to decrease as the density increased. The relative error of the self-diffusion coefficient at 1501 kg/m³ and 120 K (case 11) increased compared to that at 1400 kg/m³ and 120 K (case 14). As shown in Table 3, the absolute error of case 11 is lower than that of case 14. In contrast to the absolute error of those cases, the MD value of the self-diffusion coefficient decreases markedly, and consequently the relative error increases. This increase of the relative error does not mean the prediction performance decreases. We can speculate that it might relate to the large local density fluctuations due to the lower densities in the system. The density affects the characteristics of temporal changes in molecular velocities caused by intermolecular collision within a period corresponding to the data length used in the GANs of the present study. In other words, under a high density condition, the velocity trajectories used for the GANs considered to be relatively regular since relatively uniform intermolecular collisions occur within the period equivalent to the data length used in the GANs. However, under a low-density condition, within the period equivalent to the data length used in the GANs, molecular collisions may occur in some cases and others not, that is, non-uniform intermolecular collisions may occur. The density affects the characteristics of temporal changes in molecular velocities caused by intermolecular collision within a period corresponding to the data length used in the GANs of the present study. Additionally, Fig. S 6-9 in the Supplementary Information shows a comparison of velocity trajectories, MSD and VACF in case 8, where the data length in a single output is set to 64 and 256, respectively. These results suggested that increasing the data length for prediction improves prediction accuracy as discussed in the SI. Therefore, since the data length was constant regardless of all conditions in the present study, we can speculate that the GANs learn the characteristics of data easier under high density conditions, and the prediction accuracy by the GANs is also improved compared to the low density conditions. Also, since the GANs were trained using the batch learning method, there was a tendency to overlook samples with specific behaviors, especially in lower density conditions. We consider that also might cause the underestimation of the self-diffusion coefficient of GANs prediction compared to MD.

Figure 8 shows the relationship between temperature and self-diffusion coefficient. The relative error of the self-diffusion coefficient is shown in Fig. 9. The absolute error and relative error tended to increase as the temperature increases, and a positive correlation was suggested between the errors and temperature. The trend between the temperature and errors might be caused by the correction of domain-knowledge employed in the present study. In the present study, the domain-knowledge corrected the average and variance of molecular velocities. The variance of velocity under high-temperature conditions is larger than that under low-temperature conditions. When the variance was corrected using domain knowledge, a higher variance resulted in a greater

discrepancy between the original and generated distributions of molecular velocities. Since the variance increases with temperature, this correction may contribute to the larger error at higher temperatures. However, as shown in Fig. 6 and 8, the influence of temperature is much smaller than that of the density.

Finally, we discuss the influences of the calculation methods on self-diffusion coefficients predicted via GANs. The difference of self-diffusion coefficient calculated using method 1 and method 2 falls within the error of two significant digits as shown in Table 3. These results apparently show that the difference of the self-diffusion coefficients between MD simulations and GANs comes from the temporal trajectories of molecular velocities between MD simulations and GANs independent on the calculation procedure of methods 1 and 2. When using method 1 for the calculations of self-diffusion coefficients, it is possible to confirm the validity of the GANs prediction through qualitative trend of MSD. In contrast, the GANs can reproduce the similar trend of VACF with moving average using method 2 although they fail to reproduce the short-term fluctuations. Therefore, when evaluating in small segments of ensemble average, we consider that the method 1 is preferable for the prediction of self-diffusion coefficients rather than method 2. As we mentioned above, when we calculate the self-diffusion coefficient via method 1, we need to obtain displacement trajectories of molecules by integrating the velocity trajectory output from the generators of GANs. On the other hand, when we employ method 2, we can calculate the self-diffusion coefficient directly from the trajectory outputs of molecular velocities. The GANs employed in the present study were supposed to acquire the stochastic time evolutions of molecular velocities which were not determined by Newton's equation. Even if the velocity trajectories were ideally predicted, there was no guarantee that the integrated molecular displacements using predicted velocities would be proper to calculate the diffusion properties. However, the results obtained suggest that the integrated quantities can also be reproduced even considering the stochastic temporal evolution predicted by GANs.

The comparison of calculation time required for MD and GANs is shown in Table 4. The total calculational time of MD is smaller than that of GANs in the cases of a small number of molecules. However, for the most computationally demanding conditions (case 13-15), the MD simulations required 13 hours after relaxation calculation to obtain the 3 ns of data. In contrast, for GANs, extracting 0.9 ns of training data took about 4 hours, and the additional 6 hours were required for training and prediction, resulting in a total of 10 hours to obtain 3 ns of data. Since the processing time for GANs remains constant regardless of the conditions of MD simulation as shown in Table 4, the reduction in computational cost becomes more significant for larger and more complex systems which require long calculation time compared with the present simple and small molecular system of Lennard-Jones potentials. In the present study, all calculations were performed on the following hardware: PC : HPC5000-XIL216TS-D8, CPU : Xeon Gold 6326 (16core, 2.9 GHz and 185 W), GPU : GeForce RTX 4090 (24 GB and 450 W).

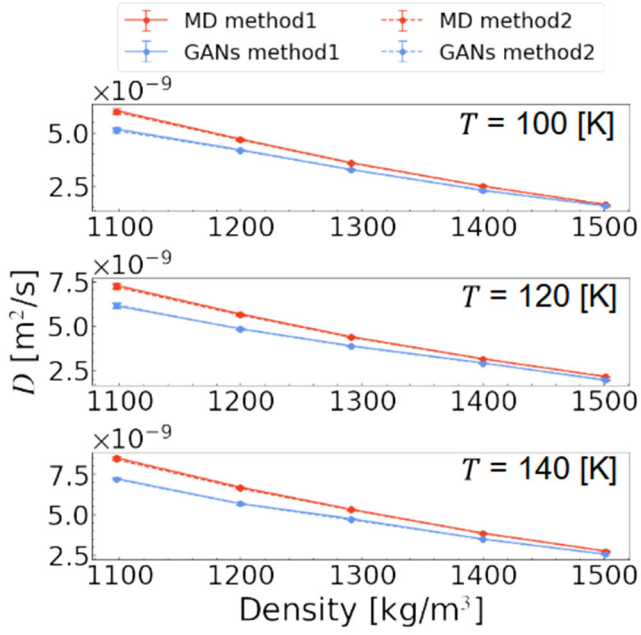


Fig. 6 Relationships between density and self-diffusion coefficient of different calculation methods (straight line is method 1, dotted line is method 2) with different samples (red is the MD sample, blue is the GANS prediction sample) at temperatures of 100, 120 and 140 K.

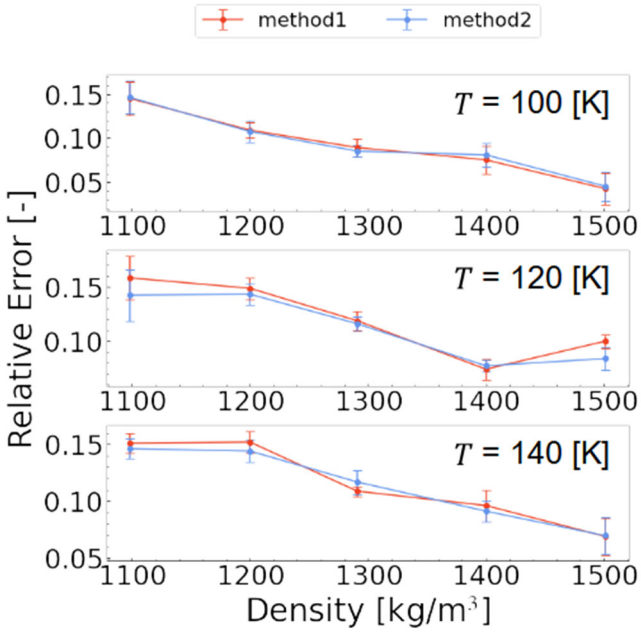


Fig. 7 Relationships between density and the relative error of the self-diffusion coefficients from different calculation methods (straight line is method 1, dotted line is method 2) at temperatures of 100, 120 and 140 K.

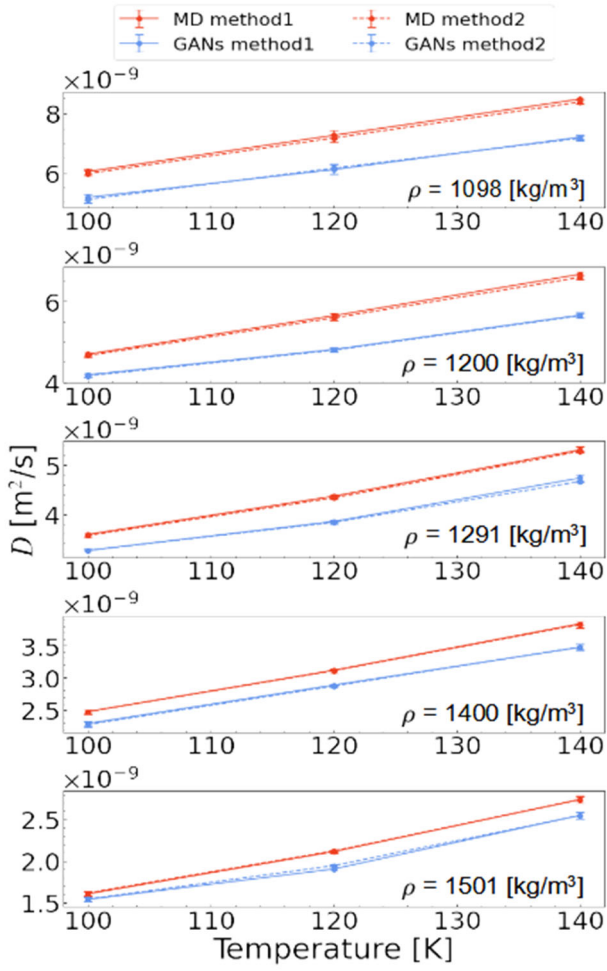


Fig. 8 Relationships between temperature and self-diffusion coefficients from different calculation methods (straight line is method 1, and dotted line is method 2) with different samples (red is the MD sample, blue is the GANS prediction sample) at densities of 1098, 1291 and 1501 kg/m^3 .

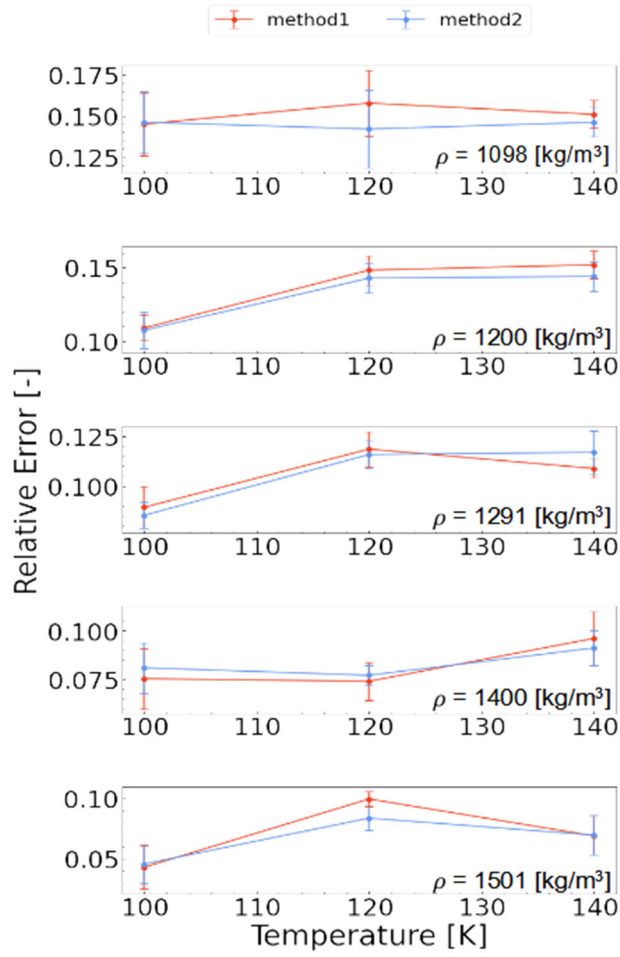


Fig. 9 Relationships between temperature and self-diffusion coefficient of different calculation methods (straight line is method 1, dotted line is method 2) at densities 1098, 1291 and 1501 kg/m³.

Table 2. Comparison of the self-diffusion coefficients found by MD simulation and GANs in each case calculated by methods 1 and 2.

	$D \times 10^{-9} \text{ [m}^2/\text{s]}$					
	GANs	MD	GANs	MD	GANs	MD
	case1		case2		case3	
method 1	5.17 ± 0.11	6.0361 ± 0.0004	6.09 ± 0.17	7.2615 ± 0.0012	7.19 ± 0.07	8.4833 ± 0.0011
method 2	5.09 ± 0.11	5.9661 ± 0.0005	6.15 ± 0.14	7.1707 ± 0.0006	7.16 ± 0.07	8.3844 ± 0.0009
	case4		case5		case6	
method 1	4.19 ± 0.06	4.6967 ± 0.0003	4.81 ± 0.06	5.6491 ± 0.0012	5.67 ± 0.07	6.6717 ± 0.0007
method 2	4.16 ± 0.04	4.6618 ± 0.0005	4.79 ± 0.06	5.5938 ± 0.0012	5.65 ± 0.06	6.5964 ± 0.0010
	case7		case8		case9	
method 1	3.27 ± 0.04	3.5853 ± 0.0006	3.85 ± 0.04	4.3675 ± 0.0005	4.74 ± 0.08	5.3149 ± 0.0007
method 2	3.25 ± 0.04	3.5626 ± 0.0009	3.83 ± 0.04	4.3333 ± 0.0004	4.67 ± 0.02	5.2846 ± 0.0004
	case10		case11		case12	
method 1	2.29 ± 0.03	2.4722 ± 0.0001	2.89 ± 0.02	3.1171 ± 0.0004	3.48 ± 0.03	3.8411 ± 0.0003
method 2	2.27 ± 0.04	2.4703 ± 0.0004	2.87 ± 0.03	3.1101 ± 0.0004	3.48 ± 0.05	3.8292 ± 0.0006
	case13		case14		case15	
method 1	1.54 ± 0.03	1.6036 ± 0.0002	1.91 ± 0.02	2.1126 ± 0.0002	2.55 ± 0.04	2.7408 ± 0.0001
method 2	1.54 ± 0.03	1.6150 ± 0.0001	1.94 ± 0.01	2.1213 ± 0.0002	2.55 ± 0.04	2.7378 ± 0.0002

Table 3. Comparison of the absolute and relative errors found by MD simulation and GANs of the self-diffusion coefficients in each case calculated by methods 1 and 2.

	Absolute error $\times 10^{-9} \text{ [m}^2/\text{s]}$	Relative error [-]	Absolute error $\times 10^{-9} \text{ [m}^2/\text{s]}$	Relative error [-]	Absolute error $\times 10^{-9} \text{ [m}^2/\text{s]}$	Relative error [-]
	case1		case2		case3	
method 1	0.88 ± 0.12	0.15 ± 0.02	1.15 ± 0.14	0.16 ± 0.02	1.28 ± 0.07	0.15 ± 0.01
method 2	0.87 ± 0.11	0.14 ± 0.02	1.02 ± 0.17	0.14 ± 0.02	1.23 ± 0.07	0.15 ± 0.01
	case4		case5		case6	
method 1	0.51 ± 0.04	0.11 ± 0.01	0.84 ± 0.06	0.15 ± 0.01	1.01 ± 0.06	0.15 ± 0.01
method 2	0.50 ± 0.06	0.11 ± 0.01	0.80 ± 0.06	0.14 ± 0.01	0.95 ± 0.07	0.14 ± 0.01
	case7		case8		case9	
method 1	0.32 ± 0.04	0.09 ± 0.01	0.52 ± 0.04	0.12 ± 0.01	0.58 ± 0.02	0.11 ± 0.004
method 2	0.30 ± 0.02	0.09 ± 0.01	0.50 ± 0.03	0.12 ± 0.01	0.62 ± 0.06	0.12 ± 0.01
	case10		case11		case12	
method 1	0.19 ± 0.04	0.08 ± 0.02	0.23 ± 0.03	0.07 ± 0.01	0.37 ± 0.05	0.10 ± 0.01
method 2	0.20 ± 0.03	0.08 ± 0.01	0.24 ± 0.02	0.08 ± 0.01	0.35 ± 0.03	0.09 ± 0.01
	case13		case14		case15	
method 1	0.07 ± 0.03	0.04 ± 0.02	0.21 ± 0.01	0.10 ± 0.01	0.19 ± 0.04	0.07 ± 0.02
method 2	0.07 ± 0.03	0.05 ± 0.02	0.18 ± 0.02	0.08 ± 0.01	0.19 ± 0.05	0.07 ± 0.02

Table 4. The comparison of calculation time required for MD and GANs.

Cases	CPU:MD(hours)	CPU/GPU:GANs(hours)
1-3	3.5	1/6
4-6	3.5	1/6
7-9	6	2/6
10-12	9	3/6
13-15	13	4/6

Conclusions

In the present study, MD simulations of liquid Ar molecules in equilibrium were performed to produce time series predictions of molecular velocities using GANs so as to investigate the applicability of GANs for the prediction of diffusion properties of liquid molecules. The impact of temperature and density conditions in the system on the GANs-based prediction method was investigated numerically, and the differences in calculated diffusion properties predicted by GANs using the Einstein and the Green–Kubo relations were found numerically. A summary of the results is as follows:

1. The probability density of molecular velocities predicted by the GANs and that sampled by MD show good agreement regardless of the density and temperature conditions.
2. It was confirmed that the prediction performance of GANs is influenced by temperature and density conditions.
3. The error in the self-diffusion coefficient predicted via GANs in comparison with that found by MD simulations tends to decrease as the density increases and the temperature decreases.
4. The self-diffusion coefficients of GANs showed no significant discrepancies between use of the Einstein and the Green–Kubo relations.

A partial discrepancy in the molecular velocity trajectory of the learning samples due to local density fluctuations might be one of the reasons for the error depending on the density. Also, the domain-knowledge correction might be one of the reasons for the error of GANs prediction depending on the temperature.

Nomenclature

D : self-diffusion coefficient	$\text{m}^2 \cdot \text{s}^{-1}$
$\mathbf{h}_j$: latent variables of integer j	
k_B : Boltzmann constant	$\text{J} \cdot \text{K}^{-1}$
l : output length of generator	
m : mass	kg
N : Avogadro constant	mol^{-1}
n_h : dimension of latent variables	
$p(A B)$: conditional probability of A when B occurred	
r : intermolecular distance	m
r_n : random number in the standard normal distribution	
s_α : standard deviation of α direction	
T : temperature	K
Δt_{GAN} : time-step of GANs output	s
Δt_{MD} : time-step of MD simulation	s
u : uniform distribution in the range of $[0,1]$	
$\mathbf{v}$: velocity	$\text{m} \cdot \text{s}^{-1}$
$v_{i,\alpha}$: velocity of α direction of the molecule i	$\text{m} \cdot \text{s}^{-1}$
$\mathbf{X}_{j:l(j+1)}$: time-series data generated by generator	
$x_{i,\alpha}$: coordinate of α direction of the molecule i	m

Subscripts

i : molecular index
j : integer of prediction loop
α : element of the three-dimensional array of (x, y, z)
1: method 1
2: method 2

References

- [1] Fitzner M, Sosso GC, Cox SJ, Michaelides A (2019) Ice is born in low-mobility regions of supercooled liquid water. *Proceedings of the National Academy of Science*, 116, 6:2009-2014. <https://doi.org/10.1073/pnas.1817135116>
- [2] Meng S, Ma X, Gong H, Wang B and Cui Z (2024) Molecular dynamics study of boiling heat transfer in nanochannels under capillary flow. *J. Molecular Liquids*, 395, 123955. <https://doi.org/10.1016/j.molliq.2024.123955>

- [3] Guevara-Carrion G, Vrabec J, Hasse H (2011) Prediction of self-diffusion coefficient and shear viscosity of water and its binary mixtures with methanol and ethanol by molecular simulation. *J. Chem. Phys.*, 134(7), 074508.
<https://doi.org/10.1063/1.3515262>
- [4] Behler J, Parrinello M (2007) Generalized neural-network representation of high-dimensional potential-energy surfaces. *American Physical Society, Phys. Rev. Lett.*, 98, 146401.
<https://doi.org/10.1103/PhysRevLett.98.146401>
- [5] Kang P-L, Shang C, Liu Z-P (2020) Large-Scale Atomic Simulation via Machine Learning Potentials Constructed by Global Potential Energy Surface Exploration. *Accounts of Chemical Research*, 53(10):2119-2129.
<https://doi.org/10.1021/acs.accounts.0c00472>
- [6] Wang Y, Lamim Ribeiro J-M, Tiwary P (2020) Machine learning approaches for analyzing and enhancing molecular dynamics simulations, *Current Opinion in Structural Biology*, 61, 139-145.
<https://doi.org/10.1016/j.sbi.2019.12.016>.
- [7] Desgranges C, Delhommelle J (2019) Ensemble Learning of Partition Functions for the Prediction of Thermodynamic Properties of Adsorption in Metal–Organic and Covalent Organic Frameworks, *J. Phys. Chem. C*, 124(3), 1907–1917
<https://doi.org/10.1021/acs.jpcc.9b07936>
- [8] Ricci E, Vergadou N (2023) Integrating Machine Learning in the Coarse-Grained Molecular Simulation of Polymers, *J. Phys. Chem. B*, 127(11), 2302-2322.
<https://doi.org/10.1021/acs.jpcc.2c06354>
- [9] Wan L-K, Xue Y-X, Jiang J-W, Park HS (2023) Machine learning accelerated search of the strongest graphene/h-BN interface with designed fracture properties. *J. Applied Phys.*, 133(2), 024302.
<https://doi.org/10.1063/5.0131576>
- [10] Yu C, Ouyang Y, Chen J (2022) Enhancing thermal transport in multilayer structures: A molecular dynamics study on Lennard-Jones solids. *Frontiers of Physics*, 17, 53507.
<https://doi.org/10.1007/s11467-022-1170-5>
- [11] Roy Chowdhury P, Reynolds C, Garrett A, Feng T, Adiga SP, Ruan X (2020) Machine learning maximized Anderson localization of phonons in aperiodic superlattices. *Nano Energy*, 69, 104428.
<https://doi.org/10.1016/j.nanoen.2019.104428>
- [12] Wang J, Li C, Shin S, Qi H (2020) Accelerated Atomic Data Production in Ab initio Molecular Dynamics with Recurrent Neural Network for Materials Research. *J. Phys. Chem. C*, 124:14838-14846.
<https://doi.org/10.1021/acs.jpcc.0c01944>
- [13] Imrie F, Bradley AR, Van der Schaar M, Deane CM (2020) Deep Generative Models for 3D Linker Design. *J. Chemical Info. and Modeling*, 60(4):1983-1995.
<https://doi.org/10.1021/acs.jcim.9b01120>
- [14] Lucio OM, Baillif B, Clevert DA, Rouquie D, Wichard J (2020) De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nature Communications*, 11, 10.
<https://doi.org/10.1038/s41467-019-13807-w>
- [15] Sase K, Shibuta Y (2023) Prediction of microstructure evolution at the atomic scale by deep generative model in combination with recurrent neural networks. *Acta Materialia*, 259, 119295.
<https://doi.org/10.1016/j.actamat.2023.119295>
- [16] Endo K, Tomobe K, Yasuoka K (2018) Multi-Step Time Series Generator for Molecular Dynamics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, 1:2192-2199.
<https://doi.org/10.1609/aaai.v32i1.11863>
- [17] Kawada R, Endo K, Yuhara D, Yasuoka K (2020) MD-GAN with multi-particle input: the machine learning of long-time molecular behavior from short-time MD data. *Soft Matter*, 18:8446-8455.
<https://doi.org/10.1039/D2SM00852A>
- [18] Kong Q, Shibuta Y, (2024) Predicting materials properties with generative models: applying generative adversarial networks for heat flux generation. *J. Phys.: Condensed Matter*, 36, 195901.
<https://doi.org/10.1088/1361-648X/ad258b>
- [19] Schütt KT, Arbabzadah F, Chmiela S, Müller KR, Tkatchenko A (2017) Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8, 13890.
<https://doi.org/10.1038/ncomms13890>
- [20] Fisher RA, Watts RO (1972) Calculated Self-Diffusion Coefficients for Liquid Argon. *Australian Journal of Physics*, 25(5):529-538.
<https://doi.org/10.1071/PH720529>
- [21] Kikugawa G, Ando S, Suzuki J, Naruke Y, Nakano T, Ohara T (2015) Effect of the computational domain size and shape on the self-diffusion coefficient in a Lennard-Jones liquid. *The Journal of Chemical Physics*, 142(2), 024503.
<https://doi.org/10.1063/1.4905545>
- [22] Kawada R, Endo K, Yasuoka K, Kojima H, Matubayasi N (2023) Prediction of Water Diffusion in Wide Varieties of Polymers with All-Atom Molecular Dynamics Simulations and Deep Generative Models. *J. Chem. Inf. Modeling*, 63(1): 76-86.
<https://doi.org/10.1021/acs.jcim.2c01316>

- [23] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville AC (2017) Improved Training of Wasserstein GANs. *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*:5769-5779.
- [24] Mescheder L, Geiger A, Nowozin S (2018) Which Training Methods for GANs do actually Converge? <https://doi.org/10.48550/arXiv.1801.04406>
- [25] Aditya NS, Jeppe CD, Ulf RP (2021) Solid-liquid coexistence of neon, argon, krypton and xenon studied by simulations. *J. Chem. Phys.*, 154, 134501
<https://doi.org/10.1063/5.0045398>
- [26] Liu P, Harder E, Berne BJ (2004) On the Calculation of Diffusion Coefficients in Confined Fluids and Interfaces with an Application to the Liquid-Vapor Interface of Water. *J. Phys. Chem. B*, 108(21):6595-6602.
<https://doi.org/10.1021/jp0375057>

Disclosure statement

The authors report there are no competing interests to declare.

Funding

This work was supported by Tanikawa Fund Promotion of Thermal Technology and the Japan Society for the Promotion of Science, “Grant-in-Aid for Science Research No. 24K21588”.

Word count

Abstract: 199 words
 Keywords: 9 words
 1. Introduction: 1095 words
 2. Methodology
 2-1. GANs system: 1108 words
 2-2. MD simulation: 322 words
 2-3. Calculation of physical properties: 333 words
 3. Results and discussion: 118 words
 3-1. Velocity distribution predicted by GANs: 157 words
 3-2. Diffusion coefficient calculated via method 1: 380 words
 3-3. Diffusion coefficient calculated via method 2: 381 words
 3-4. Comparison of MD simulation and GANs prediction: 1175 words
 4. Conclusions: 231 words
 Nomenclature: 110 words
 Subscripts: 23 words
 References: 740 words
 Disclosure statement: 10 words
 Funding: 26 words

Table 1: 57 words
 Table 2: 125 words
 Table 3: 144 words
 Table 4: 30 words

Figure captions
 Figure 1: 64 words
 Figure 2: 9 words
 Figure 3: 19 words
 Figure 4: 23 words
 Figure 5: 48 words
 Figure 6: 44 words
 Figure 7: 38 words
 Figure 8: 45 words
 Figure 9: 29 words

(Supplementary Information: 1261 words)