Accurate generation of stochastic dynamics based on multi-model Generative Adversarial Networks

Daniele Lanzoni,[1] Olivier Pierre-Louis,[2] and Francesco Montalenti[1]
[1)]*Materials Science Department, University of Milano-Bicocca, Via R. Cozzi 55, I-20125 Milano, Italy*
[2)]*Institut Lumière Matière, UMR5306 Université Lyon 1—CNRS, 69622 Villeurbanne, France*

(*Electronic mail: d.lanzoni@campus.unimib.it)

(Dated: 22 September 2023)

Generative Adversarial Networks (GANs) have shown immense potential in fields such as text and image generation. Only very recently attempts to exploit GANs to statistical-mechanics models have been reported. Here we quantitatively test this approach by applying it to a prototypical stochastic process on a lattice. By suitably adding noise to the original data we succeed in bringing both the Generator and the Discriminator loss functions close to their ideal value. Importantly, the discreteness of the model is retained despite the noise. As typical for adversarial approaches, oscillations around the convergence limit persist also at large epochs. This undermines model selection and the quality of the generated trajectories. We demonstrate that a simple multi-model procedure where stochastic trajectories are advanced at each step upon randomly selecting a Generator leads to a remarkable increase in accuracy. This is illustrated by quantitative analysis of both the predicted equilibrium probability distribution and of the escape-time distribution. Based on the reported findings, we believe that GANs are a promising tool to tackle complex statistical dynamics by machine learning techniques.

I. INTRODUCTION

Generative Adversarial Networks (GANs)[1] are a class of Machine Learning (ML) methods capable of generating data with the same statistical properties of an assigned training set. Importantly, this is accomplished without the need to explicitly estimate the target probability density, a daunting task for high-dimensional problems.

We recall that GANs are formulated[1] as an adversarial game between two neural networks (NNs), called Generator G and Discriminator D, which are trained concurrently. The task of the Generator is to map samples z extracted from a known probability distribution into samples resembling those drawn from the real (unknown) data distribution. These generated samples are passed to the Discriminator together with samples from the actual training dataset. The objective of the Discriminator is then to recognize whether its input comes from the generated or the true distribution. The two networks are trained in an alternate fashion by minimization of loss functions defining a zero-sum adversarial game. When Nash equilibrium is reached, in principle, G successfully maps z samples to elements which are distributed according to the true data distribution.

In recent years, models exploiting GANs have shown impressive results in the generation of photo-realistic images[2], text[3] and in medical imaging[4], only to cite a few applications in diverse fields. On the other hand, successful applications of GANs in statistical mechanics/computational physics seems to be very limited when compared with other ML approaches[5], which have been used, e.g., to map deterministic properties such as structure-energy relationships in molecular systems[6,7], phase classification in materials[8] and prediction of complex nonlinear dynamical evolution[9,10]. On the generative possibilities of Machine Learning approaches in physics, there are some recent applications, for example in quantum event generation[11] and in material structure and composition prediction[12,13]. Learning dynamical processes adds another layer of complexity, as it involves capturing and correctly reproducing time correlations. Nevertheless, GANs are a promising tool to model also stochastic time series data[14] and, specifically, stochastic dynamics in physical systems[15,16].

Despite their success, Generative Adversarial Networks are known to be particularly difficult to handle during training, as mode collapse and convergence failures are typically encountered[17]. Indeed, an intense activity on developing specialized architectures[18], empirical "tricks"[17,19] and theoretical understanding[20,21] has been ongoing in last years. On the dynamical systems front, the recent Ref. 15 focuses on GAN regularization techniques which allows the training of Recurrent Neural Networks capable of reproducing stochastic evolution of continuous physical systems. While alternative generative models such as diffusion denoising models[22] and variational autoencoders[23] are less prone to these convergence problems, the high quality of samples which can be achieved by GANs and their relatively fast generation times make them appealing.

In this work, we quantitatively test GANs by applying this approach to a prototypical stochastic process defined on a lattice. Convergence of the learning procedure close to Nash equilibrium and very high accuracy in the predicted dynamics is achieved upon exploiting two key procedures: noise injection and a simple, yet effective, multi-model average. These approaches are general and can be transferred to any GAN framework and architecture with straightforward modifications. While the stochastic process here considered is very simple, the reader should keep in mind that even in this one-dimensional settings, obtaining a high quality GAN can be non-trivial. On the other hand, the low dimensionality of the example, allows for an in-depth, quantitative analysis of the effect of the considered procedures. Additionally,

this is the first example, to the authors knowledge, in which a GAN is employed to learn physical dynamics coming from lattice models, which are ubiquitous in computational physics. While the authors do not expect to obtain a more efficient method for generating stochastic trajectories in this specific case, we still hope that the current work will serve as a proof of concept and will stimulate further research in tackling more complex systems, such as systems containing many interacting particles and/or more computationally expensive Kinetic Monte Carlo (KMC) variants.

The paper is organized as follows. First a stochastic process on a lattice is defined, analytical expressions for equilibrium and kinetic properties of the system and the adaptation of the standard GAN approach are reported. Next, training convergence and the effects of noise injection as a regularization procedures are discussed. In particular, we show that this technique greatly stabilize the training process and increases the quality of generated trajectories. This, however, is not enough to obtain accurate generation on the quantitative level. We therefore show that a suitable multi-model approach allows to recover quantitative agreement with close form expressions for both equilibrium and kinetic properties of the system. We conclude the Results section by outlining perspectives on transfer learning possibilities. For better clarity, we kept the most technical aspect of this work as appendices, while the main text reports results concerning optimal choices of hyperparameters.

II. METHODS

A. Stochastic process definition

In the diffusion process depicted on Fig. 1, the rate at which a particle at the site x_i moves to the right and to the left are identical and equal to

$$\gamma_i = \nu \mathrm{e}^{-(E_b - E_i)/k_B T}, \tag{1}$$

where ν is an attempt frequency, E_i is the energy of state i, E_b is the energy of the diffusion barrier, k_B is Boltzmann's constant and T is the temperature. For convenience, the lattice spacing is set to 1. Positions take therefore integer values $x_i = i$. At domain boundaries, corresponding to $x = 0$ ($x = L$) the particle is only allowed to jump to the right (left).

This diffusion process obeys detailed balance at equilibrium and the stationary probability of the Markov chain therefore corresponds to a well-defined equilibrium state with probability

$$P_{eq,i} = \frac{1}{\mathscr{Z}} \mathrm{e}^{-E_i/k_B T} \tag{2}$$

where $\mathscr{Z} = \sum_{i=1,L} \exp[-E_i/k_B T]$ is the partition function.

We chose a two-well energy profile

$$E_i = \frac{1}{2}(E_{\max} + E_{\min}) + \frac{1}{2}(E_{\max} - E_{\min}) \cos\left[\frac{4\pi i}{L}\right], \tag{3}$$

where L is the domain length.

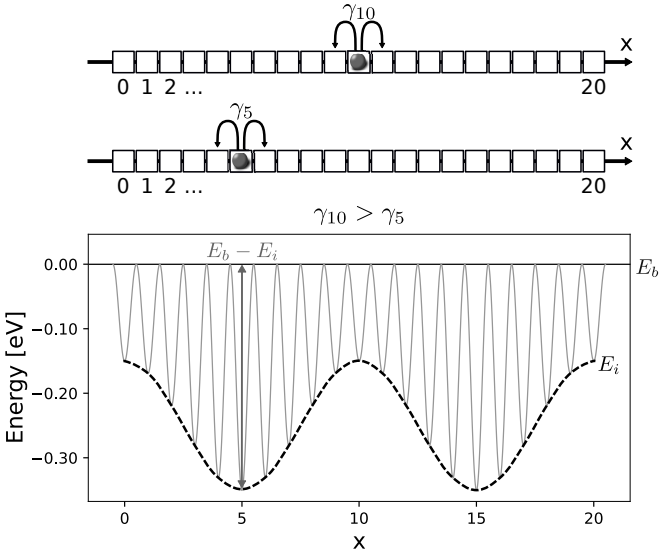

FIG. 1. Graphical representation of the random walk. Lattice points correspond to energy minima. In order to reach a new lattice position, the particle has to overcome the diffusion barrier E_b. Gray curve represent the potential profile, dashed black curve interpolates E_i.

As a representative measure of the kinetic properties of the process, we use the average time τ for going from one minimum to the other. In the continuum limit $L \gg 1$[24],

$$\tau = \frac{Ld}{2} \mathrm{e}^{-(E_{\max} + E_{\min})/2k_B T} I_0\left[\frac{E_{\max} - E_{\min}}{2k_B T}\right], \tag{4}$$

where d is the distance between the minima of the potential, and I_0 is the modified Bessel function of the first kind.

In the following, we use $L = 20$, and eV's for the energy units, with $E_{\min} = -0.35$, $E_{\max} = -0.15$, and $T = 500$K.

B. Training set

From the here-above defined stochastic process we generated independent trajectories by KMC simulations, from which we extracted the elements of the training set, i.e. $(x_t, x_{t+\Delta t})$ pairs, where x_t represents the particle position at time t and $x_{t+\Delta t}$ the position at time $t + \Delta t$.

Using time-units in which $\exp[E_b/k_B T]/\nu = 1$, we set $\Delta t = 10^3$, the mean residence times in sites with minimum and maximum energies being approximately $3\Delta t$ and $\Delta t/30$. This specific choice of Δt is not critical, except from some obvious requirements: Δt much larger than the residence times would simply yield the long-time equilibrium distribution with no access to kinetic properties. At the other extreme, values of Δt much smaller than residence times would call for too many iterations to generate meaningful trajectories.

The full training set was built upon collecting 1200 independent trajectories of 500 snapshots Δt apart from each other. The value of t itself is a multiple of Δt, so that the training set is composed of couples $\{(x_0, x_{\Delta t}), (x_{\Delta t}, x_{2\Delta t}), (x_{2\Delta t}, x_{3\Delta t}), ...\}$. In the following, the term "step" will be used for such time

intervals and not for individual Kinetic Monte Carlo steps, if not stated otherwise.

C. GAN approach

Let us now describe how GANs can be trained using this dataset. The task can be conveniently phrased in the framework of conditional GANs (cGANs)[25]. From a practical standpoint, the only difference with the original, unconditional GAN approach is that both the output of the Generator and that of the Discriminator are conditioned on some additional information. To mimic Markovian dynamics, the Generator is conditioned by the current state of the system. Hence, for assigned NN parameters θ_G, to be optimized during training, the input of G consists in the current position of the particle x_t and in the latent random variable z. Hence, we write:

$$\tilde{x}_{t+\Delta t} = G(x_t, z|\theta_G), \tag{5}$$

where $\tilde{x}_{t+\Delta t}$ is the generated next state of the system. The specific choice of the latent distribution is not crucial, provided that the dimension of z is higher[?] or equal than the one of real data $x_{t+\Delta t}$[20]. In the present work, therefore, z was conveniently sampled from a multivariate Gaussian distribution in $\mathbb{R}^{25}$ at each time step. The Generator produces real numbers and is initially unaware of both the confined and discrete characters of the stochastic dynamics (i.e. $0 \leq x_t \leq L$ and x_t takes integer values).

Similarly, the scores of the Discriminator, indicating how likely is the observation of the next state $x_{t+\Delta t}$ are conditioned on x_t:

$$\begin{aligned} s &= D(x_{t+\Delta t}, x_t|\theta_D) \\ \tilde{s} &= D(\tilde{x}_{t+\Delta t}, x_t|\theta_D) = D(G(x_t, z|\theta_G), x_t|\theta_D) \end{aligned} \tag{6}$$

where s and $\tilde{s}$ are the scores for the next state in the dataset $x_{t+\Delta t}$ and of the generated next state $\tilde{x}_{t+\Delta t}$ respectively, while θ_D are the Discriminator parameters. As in the original GAN paper[1], scores are real numbers in $(0,1)$ and the last layer of the Discriminator comprises a sigmoid activation function, as is standard practice in NN classifiers[26]. Both the Generator and the Discriminator are implemented in the PyTorch framework[27] as ResNet architectures[28] with 7 hidden layers containing 60 neurons each and have been trained using the Adam optimizer[29]. We remark that this choice of hyperparameters is not the optimal one in terms of computational costs. In the present work, however, we wanted to show that GANs can quantitatively reproduce properties of stochastic processes. This choice avoids problems related to underparametrized networks.

Training proceeds by alternative minimization of the loss functions[1]:

$$\begin{aligned} \mathscr{L}_G &= -\mathbb{E}[\log(\tilde{s})] \\ \mathscr{L}_D &= -\frac{1}{2}\left(\mathbb{E}[\log(s)] + \mathbb{E}[\log(1-\tilde{s})]\right) \end{aligned} \tag{7}$$

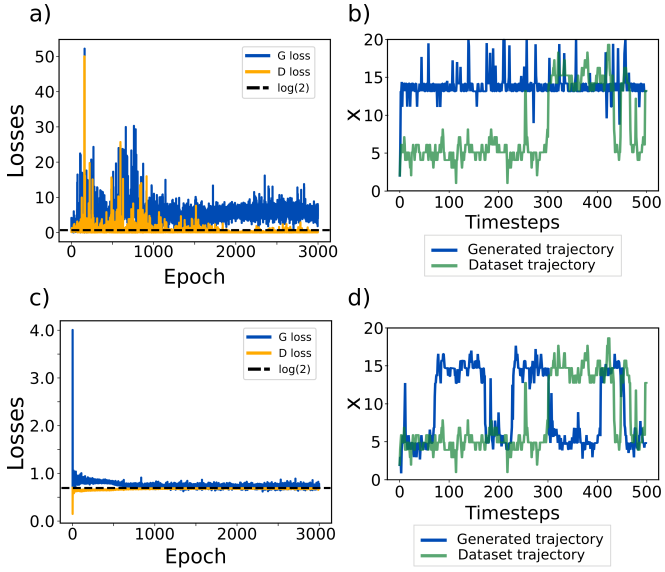

FIG. 2. (a) G and D losses as a function of the number of epochs while minimizing equations 7 directly. Theoretical Nash equilibrium value of $\log 2$ is reported for reference. The behavior is clearly non-convergent. (b) Comparison of a generated trajectory (dark blue line) at the end of procedure in (a) and one from the dataset (transparent green line). (c) Lossplot obtained when Gaussian noise ($\sigma = 0.25$) is added in the training procedure. (d) Example of a generated trajectory (dark blue line) obtained at the end of training procedure of (c) and one from the dataset (transparent green line).

where $\mathbb{E}$ is the expectation value operator. For the Generator, the non-saturating loss function discussed in Ref. 1 has been used. At Nash equilibrium, both losses in Eq. 7 are equal to $\log 2$. Pseudo-code and additional technical details on the training minimization procedure are reported in Appendix A.

III. RESULTS

A. Noise injection and training convergence

In Fig.2a we report the behavior of the Generator and Discriminator loss functions during training. Results clearly show that the training of the cGAN formulated above not only leads to oscillations, but to values of the loss functions far from the ideal $\log 2$ Nash equilibrium value. The Generator loss function exhibits strong oscillations, while $\mathscr{L}_D$ drops close to zero. This failure is also reflected in the poor quality of the generated trajectories, which can be directly assessed by visual inspection of the example reported in fig. 2b. Noticeably, the trajectory exhibits a strong tendency to oscillate only near the minimum at $x = 15$. This is reminiscent of mode collapse, one of the typical failure modes found in GANs[17,19].

This unsatisfactory result can be traced back to a feature of the original formulation of the GAN approach which is frequently encountered[17]. As pointed out by Arjovsky and Bottou[20], instabilities in training should be expected when the intersection between the support of the generated and the true

data distribution has zero measure even in the continuous case. In our example, where true data live on a discrete lattice while G outputs are real numbers, this problem is critical and results from Goodfellow et al.[1] regarding training convergence are not expected to apply.

A possible solution also proposed in Ref. 20 is based on adding random noise both to the true and generated data before passing them to the Discriminator or the Generator. Similar strategies have also been suggested as a way to reduce extrapolation errors[30] and are known to be equivalent to regularization[31]. For GANs training, this procedure expands of the support of probability distributions and solves the overlap issue. This explanation is supported by data reported in Appendix B, in which the role of additive noise was compared when training GANs on lattice and continuous dynamics.

In practice, Gaussian noise with zero mean and different standard deviations σ were added to both true and generated examples:

$$\begin{aligned} &x_{\text{data}} \rightarrow x_{\text{data}} + \varepsilon_{\text{data}} \\ &G(x_{\text{data}}, z) \rightarrow G(x_{\text{data}} + \varepsilon_{\text{data}}, z) + \varepsilon_{\text{gen}} \end{aligned} \tag{8}$$

where x_{data} represent any instance of the particle position from the database. Random variables $\varepsilon_{\text{data}}$ and ε_{gen} come from the same distribution, but are indicated with different symbols in order to stress their independence. Every time particle coordinates are passed to either the Generator or Discriminator, new random noise is added. Notice that random noise has to be added also to generated samples. This is in order to avoid that the Generator learns a degenerate distribution, i.e. its output is a single real number. This would restore the zero-measure problem and make the training procedure unstable again[20].

Increasing the value of σ increases the speed of convergence and decreases the fluctuations around Nash equilibrium. However, when σ becomes too large, the spatial distribution of the position convolved with the noise becomes smooth and the information associated with the discreteness of the lattice positions is lost. In our case, $\sigma = 0.25$ produced the best trajectories. A detailed analysis as a function of σ is reported in Appendix C. The loss plot for $\sigma = 0.25$ is reported in figure 2c. The loss functions are now approaching the ideal value, despite some oscillations being still present (notice the smaller y-axis scale in 2c as compared to 2a).

During trajectory generation, no noise was added to the Generator output. Additive noise for G inputs, however, has been retained, consistently with the training procedure. The quality of the generated trajectories is greatly increased with respect to the noiseless minimization, as seen in the example of fig. 2d. Remarkably, despite the smoothing role of ε in training, particle positions are closely peaked near discrete x values. Additionally, it is clear that the mode collapse problem encountered in the naive minimization approach has been strongly reduced. Both features has been further analyzed in Appendix D.

B. Multi-model generation of trajectories

As already pointed out, oscillations of the loss functions of Fig. 2c are persistent. This suggests that Gs extracted at different epochs within the Nash equilibrium regime could exhibit significant differences. We therefore compared equilibrium and kinetic properties of the trajectories produced by several Generators with those obtained by our reference KMC simulations. Equilibrium distributions from 4 different Generators extracted within the Nash equilibrium regime and separated by 50 epochs from each other are reported in 3a. They are obtained from 10 independent generated trajectories reaching a total time of $5 \times 10^5 \times \Delta t$. Each single G appears to produce qualitatively satisfactory results. Indeed, the expected two-peak equilibrium distribution is obtained in all cases, and as reported in Fig. 2 each Generator produces trajectories that looks reasonable. However, significant quantitative variability is found in the distributions obtained by different Gs. For instance, the height and shape of the two peaks is different at every epoch. Furthermore, each generated distribution is different from the true distribution extracted from KMC simulations, shown in Fig.3b.

Let us now show that this issue can be conveniently addressed by a multi-model scheme based on a suitable average which strongly increases the accuracy of predictions. This is done by randomly selecting Generators obtained at different epochs. Combining multiple models is often exploited in Machine Learning to increase their quality. Methods such as bagging[32], boosting[33] and stacking[34] represent only a handful of classical examples[35]. In adversarial contexts, the use of model ensembles has been proposed during training in order to make models more resilient to attacks[36]. The key issue in our case is that conditional probabilities are not explicitly accessible for a direct average. In order to circumvent this problem, we use the following procedure to generate a trajectory. Let us consider a set of models, each one associated with a different Generator extracted in the Nash equilibrium regime. At each time step Δt, we select randomly a model j from the set composed by M models. Invoking the law of total probability[37], the multi-model conditional probability $P_{\text{mm}}(x_{t+\Delta t}|x_t)$ is found to be equal to the average conditional probability:

$$\begin{aligned} P_{\text{mm}}(x_{t+\Delta t}|x_t) &= \sum_j^M P_j(x_{t+\Delta t}|x_t) P(j) \\ &= \frac{1}{M} \sum_j^M P_j(x_{t+\Delta t}|x_t) \end{aligned} \tag{9}$$

were $P_j(x_{t+\Delta t}|x_t)$ is the conditional probability for model j, $P(j) = 1/M$ is the (uniform) probability that model j is chosen.

C. Equilibrium distribution

Figure 3c shows the equilibrium distribution as obtained from the aforementioned averaging procedure using Gs at the end of the last 300 epochs of training. Further discussions

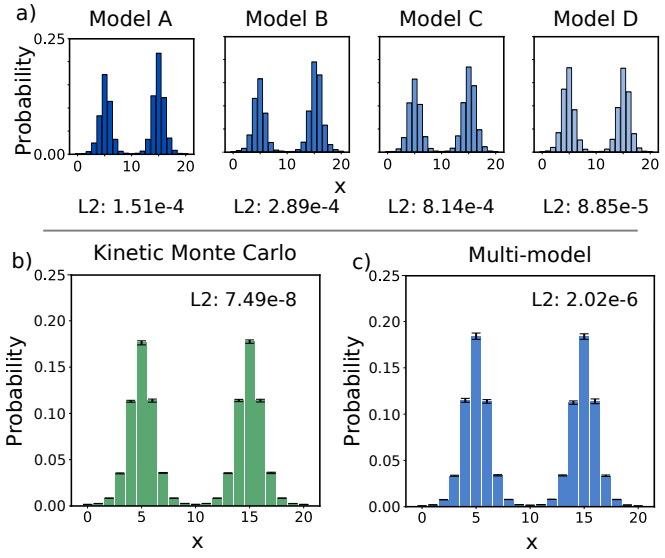

FIG. 3. Equilibrium distribution obtained by random walk trajectories from different models. Models A, B, C and D in (a) refer to individual models picked at different epochs. (b) reports the equilibrium distribution obtained by Kinetic Monte Carlo simulations. (c) has been obtained by the multi-model approach. Black error bars represent confidence intervals (not reported in (a)). L2 distances between equilibrium distributions and the analytical one are also reported.

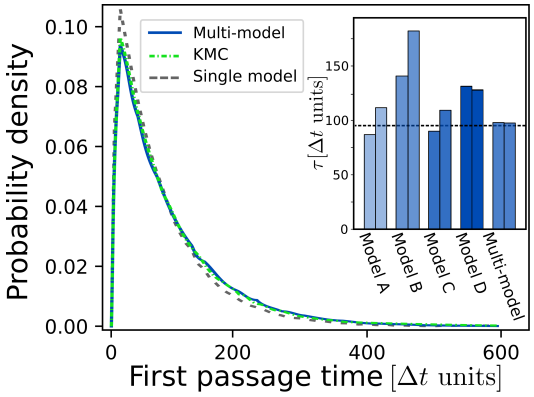

FIG. 4. First passage time distribution obtained by generated trajectories (dark blue line), by KMC simulations (dashed-dotted green line) and by a Model A (dashed gray line). Inset reports first passage times as predicted by models at different epochs and by the multi-model approach. Models A, B, C and D are the same of figure 3. Black dashed line corresponds to the analytical mean value. Values for passages from the left (right) minimum to the right (left) one are reported separately.

on the choice of the number of models and how they are extracted are reported in Appendix E. The distribution obtained in this way is remarkably close to that obtained by direct KMC simulations. Quantitatively, the L2 distance to the analytical equilibrium distribution drops by 2 orders of magnitudes, as reported in the insets of 3.

D. Kinetic properties

Using the same multi-model procedure, we also generated 10^5 independent trajectories and registered the time required to reach the left (right) minimum for a particle starting from the right (left) one. Distributions of first passage times from one minimum to the other, obtained from KMC simulations, the multi-model approach and one individual model, are reported in Fig. 4. KMC and multi-model distributions show remarkable quantitative agreement, confirming that the proposed method is indeed capable of providing an accurate description of kinetic properties of the system.

The inset of Fig. 4 shows how the average first passage time τ changes with the multi-model procedure. Passage times for reaching the right minima from the left one and vice versa are reported separately. The theoretical expression in the continuum limit reported above predicts $\tau \approx 95.14$, while direct estimation from KMC simulations give $\tau \approx 95.2$. The multi-model approach recovers these values within a 3% error.

Notice that the multi-model approach reported here is more powerful than running every model in parallel and then taking average properties, since it is capable of generating more realistic individual trajectories, which allows for the analysis of kinetic pathways. We also remark that this approach is not computationally demanding, as all models are obtained during a single training procedure.

E. Transfer learning perspectives

Finally, we wish to emphasize that GANs could conveniently be used for transfer learning. Indeed, as shown in Appendix F, a re-training of only a few epochs is sufficient to reach a new Nash equilibrium when the diffusion potential is changed slightly. However, longer training might be necessary for important changes in the dynamics.

Moreover, the transient deviation of the loss during re-training, which occurs only if the new samples have different statistical properties from the reference process, could be used to discriminate the original stochastic process from other stochastic processes. A few preliminary tests in this direction are also reported in Appendix F. This opens novel directions to detect changes as compared to a reference stochastic process.

IV. CONCLUSIONS

In conclusion, we have shown that Generative Adversarial Network architectures can be fruitfully applied to learn stochastic dynamics and are capable of reproducing both equilibrium and kinetic properties of a stochastic physical system. In addition, we have illustrated how GANs can offer promising perspectives for transfer-learning and discrimination of stochastic processes.

As compared to previous attempts to learn stochastic processes based on GANs, our method does not rely on maximum mean discrepancy regularization[15,16]. Furthermore, we

have proposed a multi-model procedure which, coupled to simple noise-regularization, allows one to achieve quantitative learning of the target stochastic process. This procedure shares similarities with usual ensemble-learning strategies[36,38,39]. However, our multiple models differ from usual ensemble learning strategies based on multiple training since they are extracted from a single learning run within the regime where the system fluctuates around Nash equilibrium. Additionally, our multi-model procedure differs in the sense that the average on the generators ensemble is taken as the system evolves in time, while in typical applications multiple models concur to a single prediction.

We have applied this procedure to the simple problem of diffusion in a potential, analyzing on quantitative grounds the outcomes of the learning procedure. We find it intriguing to notice that the present strategy could in principle be applied directly to experimental data, provided that a sufficiently large set of observations can be collected.

The next obvious step is the extension to several dimensions. While, in general, the versatility of Neural-Network based Machine-Learning methods should allow for a direct extension, particular care is needed in devising training strategies keeping the number of parameters under control. While in some cases we already obtained promising preliminary results, a complete generalization still requires further work.

ACKNOWLEDGMENTS

F.M. acknowledges financial support from ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU".

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding authors upon reasonable request.

Appendix A: Training Procedure

In this section, we provide more details on the training procedure employed for the minimization of the adversarial Loss functions:

$$\begin{aligned} \mathscr{L}_G &= -\,\mathbb{E}[\log(\tilde{s})] \\ \mathscr{L}_D &= -\,\frac{1}{2}\big(\mathbb{E}[\log(s)] + \mathbb{E}[\log(1-\tilde{s})]\big) \end{aligned} \tag{A1}$$

where $\mathscr{L}_G$ and $\mathscr{L}_D$ are the Generator and Discriminator loss functions respectively, s and $\tilde{s}$ are the scores provided by the Discriminator and $\mathbb{E}$ represent the expectation value operator. Minimization has been performed using standard PyTorch implementation[27] of the Adam optimizer[29], with batches composed of ≈ 7000 coordinates pairs. Adam parameters have been set to $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate was 10^{-4}. Pseudocode for the optimization is reported in Algorithm 1. As the Discriminator should be at optimality for a fixed Generator[1], 20 $\mathscr{L}_D$ minimization steps were performed for every $\mathscr{L}_G$ minimization step. In our tests, this produced the best trade off between quality of the generated trajectories and training time.

```
tot_epochs ← set number of epochs ;
σ ← set additive noise standard deviation ;
G ← initialize Generator ;
D ← initialize Discriminator ;
for epoch in tot_epochs do
    for (x_t, x_{t+Δt}) in dataset do
        (ε_1, ε_2) ← samples from N(0, σ) ;
        x_t ← x_t + ε_1 ;
        x_{t+Δt} ← x_{t+Δt} + ε_2 ;
        for 20 iterations do
            z ← sample from N(0, I_25) ;
            ε_3 ← sample from N(0, σ);
            x̃_{t+Δt} ← G(x_t, z) ;
            x̃_{t+Δt} ← x̃_{t+Δt} + ε_3 ;
            s̃ ← D(x̃_{t+Δt}, x_t) ;
            s ← D(x_{t+Δt,x_t}) ;
            L_D ← −½ [log(s) + log(1 − s̃)] ;
            Adam update for θ_D ;
        end
        s̃ ← D(x̃_{t+Δt}, x_t) ;
        L_G ← −log(s̃) ;
        Adam update for θ_G;
    end
    Save G and D
end
```

Algorithm 1: Algorithm used to minimize eq. A1. **for** loop iterating on the dataset operates on batches. $\mathscr{N}(0, I_{25})$ indicates a multivariate normal distribution in $\mathbb{R}^{25}$ having the identity as covariance matrix, while $\mathscr{N}(0,\sigma)$ represents the standard normal distribution with standard deviation σ.

Appendix B: Noise regularization and discrete stochastic dynamics

The stabilization effect of additive noise during training is a critical step whenever the stochastic dynamics to be learned lives on a discrete lattice. This statement follows Goodfellow et al.[26], who claimed in their original paper that "GANs require differentiation through the visible units, and thus cannot model discrete data". This is also consistent with the work of Arjovsky and Bottou[20], which shows that GANs training stability depends on the intersection conditions of the true data and generated probability distribution supports.

As a test, we trained GANs on two prototypical stochastic processes: a Gaussian step and a discrete step one-dimensional random walk without any external potential. In both cases, a particle performs an unbiased one-dimensional random walk on the interval $[0, 20]$. In the first model, at every timestep, a standard normal random variable is added to the

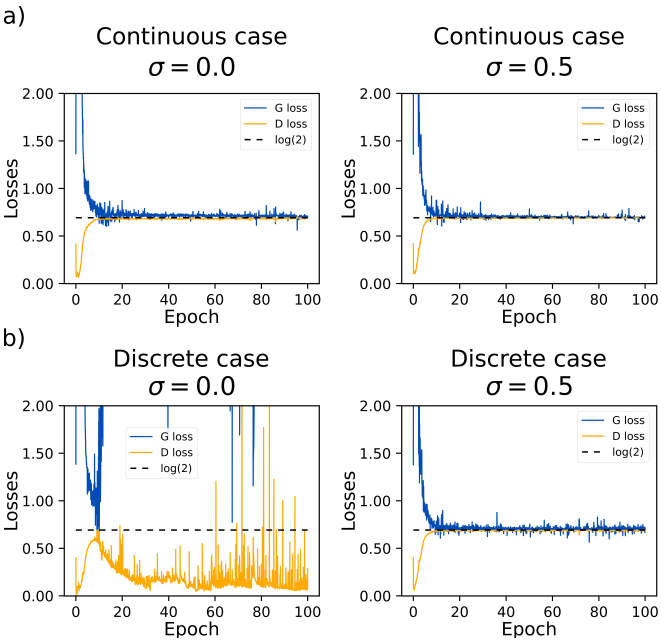

FIG. 5. (a) Lossplot obtained by training a GAN on trajectories obtained by the random walk on continuous values. (b) Reports the same loss functions in the case of training on discrete dynamics.

position of the particle. In the second model, the particle takes instead a unit step to the left or to the right with equal probabilities. Motion is confined in $[0,20]$ by rejecting moves which would place the particle outside such interval. Datasets for both the continuous and the discrete model were constructed, each comprising $(x_t, x_{t+\Delta t})$ pairs coming from 200 independent trajectories composed by 10^4 random walk steps. In order to mimic the procedure reported in the main text, in which the time interval between generated states can be bigger than individual diffusion steps, pair elements were spaced by a Δt corresponding to 10 KMC steps.

Fig. 5a reports the training loss function from a *G-D* couple with the same hyper-parameters as in the main text trained on $(x_t, x_{t+\Delta t})$ coming from the Gaussian walk model. While noise regularization slightly reduces oscillations, it does not seem to be critical for the continuous case, as both loss functions rapidly converge towards the theoretical $\log 2$ value. On the other hand, the $\sigma = 0.0$ case for the discrete dynamics shows a clearly non-convergent behavior (Fig. 5b). This is readily solved when additive noise is introduced in the training procedure.

The noise-regularization procedure, therefore, seems to be critical for model convergence when the GAN approach used in the present work is applied to discrete stochastic systems. In continuous systems, on the other hand, it plays a less crucial role but still allows for a reduction in loss functions oscillations around the Nash value $\log 2$. As a side note, we remark that applications to continuous dynamics, while not fully explored in the current work, seem to be less affected by training stability problems.

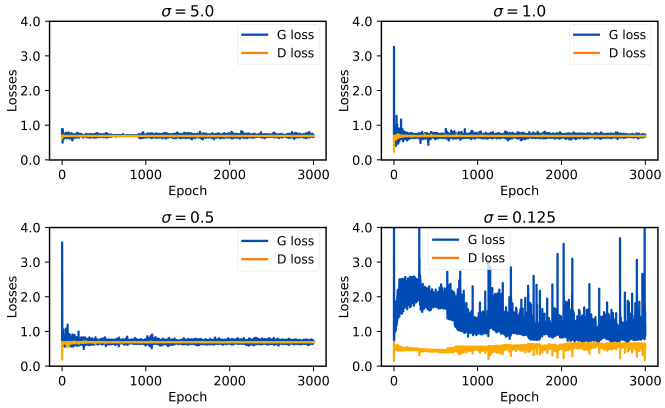

FIG. 6. Lossplot for different values of the training noise σ. The lower the standard deviation of the additive noise, the lower is the stabilization effect.

Appendix C: Training results as a function of noise strength

In the main text, we discussed results for a noise standard deviation of $\sigma = 0.25$, claiming that, among tested values, it provides the best results in terms of training convergence speed and quality of the generated trajectories. We report in Fig. 6 the lossplots for other values of σ.

As the value of σ decreases, the oscillations in loss functions during training become stronger. In the $\sigma = 0.125$ case oscillations are so strong that the 3000 epochs used here are not sufficient to conclude if convergence can be reached.

These tests show that higher variance for ε leads to more stable training procedures. Notice, however, that this does not suffice for σ selection, as lossplots do not contain quantitative information on the quality of the generated trajectories.

Appendix D: Quality of generated trajectories as a function of noise strength

We now report an analysis of the trajectories produced by training GANs with non-optimal values of σ. Fig. 7 shows the equilibrium distributions obtained for the same values as those used in the previous section. These distributions have been obtained through the same procedure as that reported for $\sigma = 0.25$ in the main text, using the multi-model approach.

It can be noticed that at high σ values details on the equilibrium distribution are lost. For example, for $\sigma = 5.0$ the Generator is merely capable of recovering the most basic aspect of the dynamics, i.e. the presence of two peaks in the equilibrium distribution. In contrast, for $\sigma = 0.125$, the probability density is suppressed in regions close to potential energy maxima, i.e. configurations that are rarely visited.

These learning failures can also be observed in the mean passage times, reported in the bottom panel of Fig. 7, where the values for $\sigma = 0.25$ are also reported for reference. Strong noise leads to a spurious reduction in learned passage times, and $\sigma = 0.125$, on the other hand, exhibits significantly longer τ.

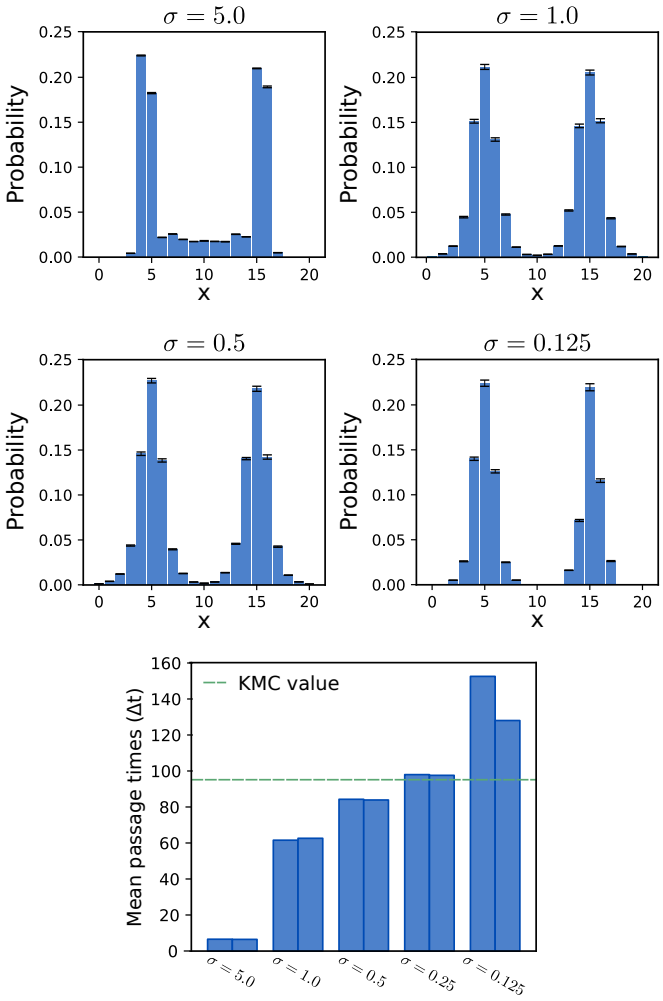

FIG. 7. Equilibrium distributions obtained by Generators trained with different additive noise strength. Black bars represent confidence interval for probabilities. Barplot on the bottom reports mean passage times. For every σ value, the two bars represent values for left to right and right to left passages respectively. KMC estimate of mean passage time is reported as a green dashed line.

Noise strength has an additional effect on the generated trajectories: the lower the noise standard deviation, the more discrete-like the trajectories appear. This property can be appreciated in Fig. 8, where the generated equilibrium distributions P_{eq} as in 7 are plotted in dark blue with smaller discretization bins (0.04 in the units of lattice spacing). Probability densities are rescaled for clarity. For $\sigma = 5.0$ (not reported), $\sigma = 1.0$ and $\sigma = 0.5$ the learned probability density is spread out into two broad peaks, corresponding to the two minima in the potential energy function. On the other hand, the two cases with $\sigma = 0.25$ and $\sigma = 0.125$ exhibit a comb-like distribution, with sharp peaks at each lattice site.

This could be explained by considering how the additive noise affects the Nash equilibrium condition for GAN training. Following Ref. 20, the Generator task is now to produce samples $\tilde{x}_{t+\Delta t}$ such that, once noise ε is added, are distributed as $x_{t+\Delta t} + \varepsilon$. Close to Nash equilibrium, therefore, Generator

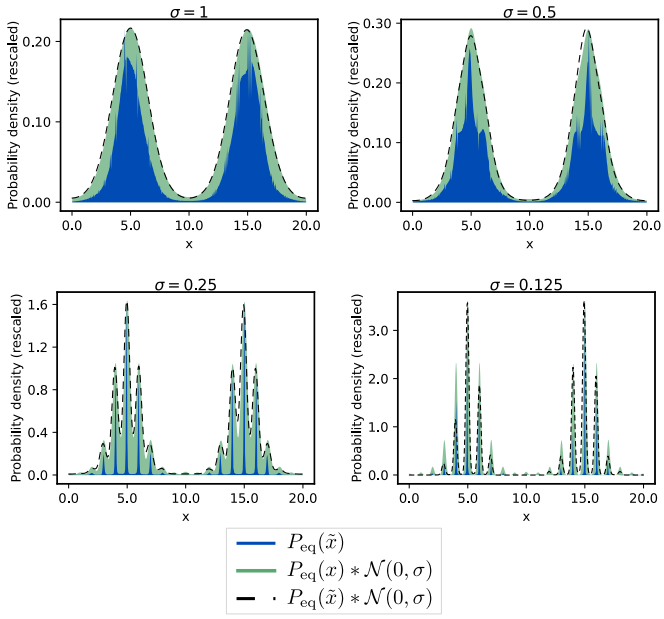

FIG. 8. Equilibrium distributions obtained by KMC convolved with the noise distribution ($P_{\text{eq}}(x) * \mathcal{N}(0,\sigma)$, pale green), and by generators, both with ($P_{\text{eq}}(\tilde{x}) * \mathcal{N}(0,\sigma)$, dark blue) and without convolution ($P_{\text{eq}}(\tilde{x})$, dashed black line).

and true data (conditional) probability densities, P_{g} and P_{data}, should be related via

$$P_{\text{g}}(\tilde{x}_{t+\Delta t}|x_t) * \mathcal{N}(0,\sigma) \approx P_{\text{data}}(x_{t+\Delta t}|x_t) * \mathcal{N}(0,\sigma), \quad \text{(D1)}$$

where $*$ denotes the standard convolution product and $\mathcal{N}(\mu,\sigma)$ a normal distribution with mean μ and standard deviation σ.

In Fig. 8, equilibrium distributions convolved with the respective $\mathcal{N}(0,\sigma)$ are reported (pale green). Clearly, information about the lattice discreteness is completely lost for large σ, which prevents learning microscopic details on valid particle positions. However, the training converges to the modified Nash equilibrium of eq. D1, as confirmed by the observation that the convolution of the generated equilibrium distribution $P_{\text{eq}}(\tilde{x}) * \mathcal{N}(0,\sigma)$ represented by the dashed black lines in 8 is remarkably close to the target one $P_{eq}(x) * \mathcal{N}(0,\sigma)$. We also remark that for $\sigma = 0.25$, despite the sharp nature of the peaks in the probability distribution, there is no significant mode collapse, as all lattice position are visited. On the contrary, for other values of σ, a depletion of probability density in energy-maxima regions can be clearly observed. In summary, these results show how poor trajectories and good training convergence can coexist. One therefore should not rely solely on $\mathscr{L}_G$ and $\mathscr{L}_D$ values.

This analysis also shows how the optimal value of σ comes from the trade off between training stabilization and the loss of microscopic details of the dynamics. Fig.8 suggests that σ should be small enough for the equilibrium distribution P_{eq} convoluted by the noise distribution $\mathcal{N}(0,\sigma)$ to exhibit one peak per lattice site, and large enough for these peaks to have overlapping tails between the peaks. The first condition is necessary to keep the information that the lattice discreteness is

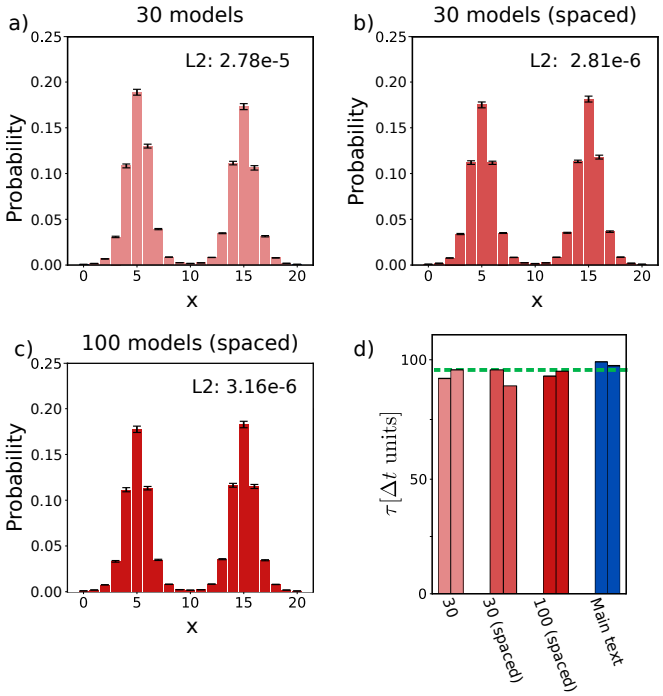

FIG. 9. Results obtained through the multi-model approach by using different models for the average. (a), (b) and (c) represent the equilibrium distribution obtained using Generators at the end of the last 30 epochs, by selecting 1 Generator every 10 in the last 300 epochs and 100 models spaced by 10 epochs in the last 1000. (d) reports mean first passage times (main text bars are reported for comparison; left to right and right to left passages are reported separately). KMC value is reported as a dashed green line.

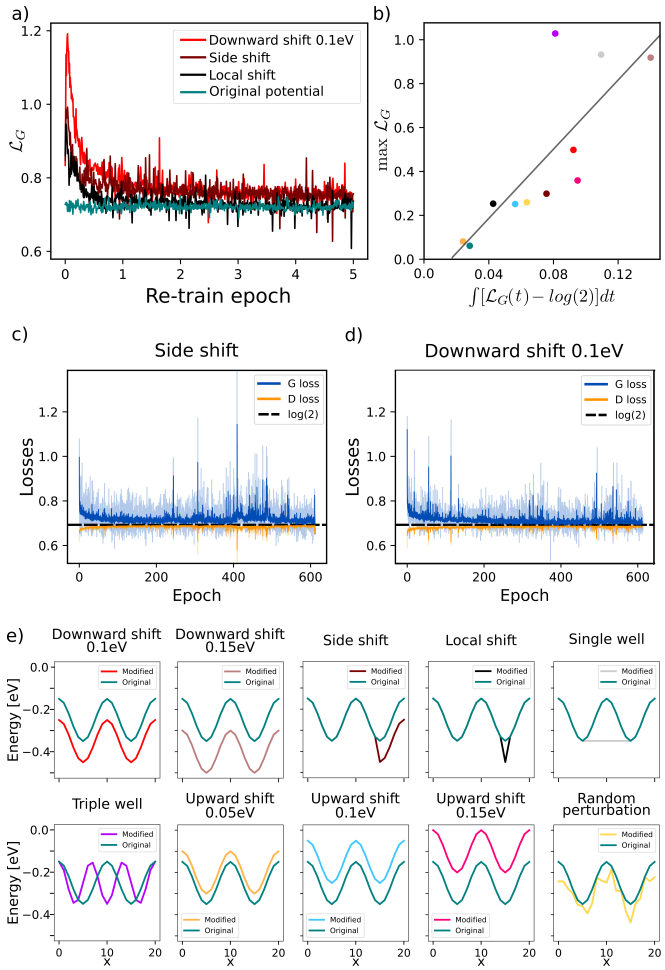

FIG. 10. (a) Loss-plot for re-training with different potentials. Perturbations correspond to a downward shift of the potential by 0.1eV for all points ("Downward shift 0.1eV"), for points at $x \geq 15$ ("Side shift") and for the point a t $x = 15$ ("Local shift"). The teal curve "Original potential" has been obtained from new trajectories coming from the original potential of fig. 1 of the main text. (b) Correlation plot between the peak height and the integral of the loss function over five epochs. Pearson correlation coefficient is ≈ 0.801. The points correspond to the 10 potentials reported in (e), and the green point corresponds to re-training with a new data set produced with the initial potential. (c,d) Longer re-trainings. (e) Perturbed potentials.

present for the learning process. The second condition guarantees that the distribution is smooth and finite everywhere, allowing e.g. for the calculation of gradients. We therefore expect that the lattice spacing should correspond to at least some units of σ. Further tests on more general lattices and higher dimensional cases, however, might exhibit differences.

Appendix E: Choice of the number of Generators in the multi-model approach

In this section we provide more details on the effect of the number of Generators used in the multi-model approach. In the main text, NN models at the end of the last 300 training epochs were chosen, as the simple architecture considered in this work does not pose memory or computational constraints.

Fig. 9, shows equilibrium distribution and mean passage times with different ways to choose the Generators for the multi-model method: the last 30 Generators in the training, 30 Generators spaced by 10 epochs in the last 300 and 100 Generators spaced by 10 epochs in the last 1000. The first choice (Fig. 9a) leads to worst quality in the generated trajectories, but is still an improvement with respect to the single model prediction. The 30 spaced models (Fig. 9b) performs very similarly to last 300 used in the main text, and provides results that are also similar to 100 spaced Generators in terms of L2 distance with the analytical distribution. Moreover, as seen from 9, there is no significant changes in the accuracy of the first passage times with our different numbers of models. In summary, the best tradeoff in terms of computational cost and accuracy among those that we have tested is to use 30 Generators spaced by 10 epochs, while the 300 models choice reported in the main text produces slightly more accurate results.

Appendix F: Re-training for transfer learning and discrimination

In this section, we propose to re-train GANs with a new KMC trajectory dataset after a converged training on the original dataset. The new dataset is produced by diffusion in a modified potential. Our aim is to investigate possible perspectives of using GANs for transfer-learning or for discrimination between the original dataset and datasets produced by other stochastic processes.

The re-training loss $\mathscr{L}_G$ is reported in Fig. 10a for four different datasets. Perturbations of the potential were performed by a downward shift of the potential by $0.1eV$, respectively for the whole potential ("Downward shift 0.1eV" in 10(e)), for lattice points at $x \geq 15$ ("Side shift" in 10(e)) and for the single point at $x = 15$ ("Local shift" in 10(e)). An additional curve for new trajectories coming from the same potential ("Original potential") is also reported. For stronger perturbations in the potential, the initial peak in the loss function and the area between $\mathscr{L}_G$ and $\log 2$ are larger. While the loss relaxes to the Nash equilibrium value after a few epochs for the local shift, the relaxation is slower for larger changes of the potential ("Global shift" and "Side shift"). 10b,c reports the losses obtained during a longer training. The relaxation towards Nash equilibrium is faster than the initial training (see Fig.3 of the main text), but is of the same order of magnitude. These results suggest that small perturbations of the potential can be learnt quickly and open interesting directions for future research on transfer learning.

Re-training also opens perspectives in the discrimination between different stochastic processes. Indeed, as a byproduct of the training procedure, the Discriminator should in principle contain implicit information on the distribution of the Generator and true data. However, near convergence the output of the Discriminator should be close to $1/2$. Therefore, we have not been able to use it directly to identify whether a new dataset comes from dynamics on the same potential or has been produced by a different stochastic process. However, since it leads to a deviation of the losses from the Nash equilibrium value, re-training does distinguish between different datasets, as shown in 10. Furthermore, the deviation of the losses occurs already in the first epochs of retraining. Hence, distinguishing between the original dataset and other datasets can be done quickly even if retraining has not converged to Nash equilibrium. As an additional supporting data for the fact that the initial peak of the loss contains robust information about the full relaxation during longer retraining, 10b shows that the peak value is correlated with the integrated deviation of the loss from the Nash equilibrium value over 5 epochs.

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Vol. 27, edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Curran Associates, Inc., 2014).

[2] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *International Conference on Learning Representations* (2018).

[3] L. Yu, W. Zhang, J. Wang, and Y. Yu, "SeqGAN: Sequence generative adversarial nets with policy gradient," Proceedings of the AAAI Conference on Artificial Intelligence **31** (2017), 10.1609/aaai.v31i1.10804.

[4] X. Yi, E. Walia, and P. Babyn, "Generative adversarial network in medical imaging: A review," Medical Image Analysis **58**, 101552 (2019).

[5] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," Rev. Mod. Phys. **91**, 045002 (2019).

[6] T. Mueller, A. Hernandez, and C. Wang, "Machine learning for interatomic potential models," The Journal of Chemical Physics **152**, 050902 (2020), _eprint: https://doi.org/10.1063/1.5126336.

[7] P. Friederich, F. Häse, J. Proppe, and A. Aspuru-Guzik, "Machine-learned potentials for next-generation matter simulations," Nature Materials **20**, 750–761 (2021).

[8] H. W. Chung, R. Freitas, G. Cheon, and E. J. Reed, "Data-centric framework for crystal structure identification in atomistic simulations using machine learning," Phys. Rev. Mater. **6**, 043801 (2022).

[9] K. Yang, Y. Cao, Y. Zhang, S. Fan, M. Tang, D. Aberg, B. Sadigh, and F. Zhou, "Self-supervised learning and prediction of microstructure evolution with convolutional recurrent neural networks," Patterns **2**, 100243 (2021).

[10] D. Lanzoni, M. Albani, R. Bergamaschini, and F. Montalenti, "Morphological evolution via surface diffusion learned by convolutional, recurrent neural networks: Extrapolation and prediction uncertainty," Phys. Rev. Mater. **6**, 103801 (2022).

[11] S. Otten, S. Caron, W. de Swart, M. van Beekveld, L. Hendriks, C. van Leeuwen, D. Podareanu, R. Ruiz de Austri, and R. Verheyen, "Event generation and statistical sampling for physics with deep generative models and a density information buffer," **12**, 2985 (2021), publisher: Nature Publishing Group UK London.

[12] S. Kim, J. Noh, G. H. Gu, A. Aspuru-Guzik, and Y. Jung, "Generative adversarial networks for crystal structure prediction," **6**, 1412–1420 (2020), publisher: ACS Publications.

[13] Y. Dan, Y. Zhao, X. Li, S. Li, M. Hu, and J. Hu, "Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials," **6**, 84 (2020), publisher: Nature Publishing Group UK London.

[14] E. Brophy, Z. Wang, Q. She, and T. Ward, "Generative adversarial networks in time series: A systematic literature review," ACM Comput. Surv. **55** (2023), 10.1145/3559540.

[15] K. Yeo, Z. Li, and W. Gifford, "Generative adversarial network for probabilistic forecast of random dynamical systems," **44**, A2150–A2175 (2022), place: USA Publisher: Society for Industrial and Applied Mathematics.

[16] P. Stinis, C. Daskalakis, and P. J. Atzberger, "SDYN-GANs: Adversarial learning methods for multistep generative models for general order stochastic dynamics," (2023), https://doi.org/10.48550/arXiv.2302.03663, _eprint: 2302.03663.

[17] D. Saxena and J. Cao, "Generative adversarial networks (GANs): Challenges, solutions, and future directions," ACM Comput. Surv. **54** (2021), 10.1145/3446374.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," (2015).

[19] I. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," (2017), _eprint: 1701.00160.

[20] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," (2017), https://doi.org/10.48550/arXiv.1701.04862, _eprint: 1701.04862.

[21] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of GANs," **30** (2017).

[22] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," (2020), _eprint: 2006.11239.

[23] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," (2022), _eprint: 1312.6114.

[24] N. G. Van Kampen, *Stochastic processes in physics and chemistry*, Vol. 1 (Elsevier, 1992).

[25] M. Mirza and S. Osindero, "Conditional generative adversarial nets," arXiv preprint arXiv:1411.1784 (2014).

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT press, 2016).

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. De-

Vito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019).

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980 (2014), https://doi.org/10.48550/arXiv.1412.6980.

[30] P. Stinis, T. Hagge, A. M. Tartakovsky, and E. Yeung, "Enforcing constraints for interpolation and extrapolation in generative adversarial networks," Journal of Computational Physics **397**, 108844 (2019).

[31] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," Neural Computation **7**, 108–116 (1995).

[32] L. Breiman, "Bagging predictors," Machine Learning **24**, 123–140 (1996).

[33] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of online learning and an application to boosting," Computational Learning Theory , 23–37 (1995).

[34] D. H. Wolpert, "Stacked generalization," Neural Networks **5**, 241–259 (1992).

[35] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," Engineering Applications of Artificial Intelligence **115**, 105151 (2022).

[36] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," (2020), _eprint: 1705.07204, _eprint: 1705.07204.

[37] R. B. Ash, *Basic probability theory* (John Wiley & Sons, Inc., 1970).

[38] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," Frontiers of Computer Science **14**, 241–258 (2020).

[39] Z.-H. Zhou, "Ensemble learning," in *Machine Learning* (Springer Singapore, Singapore, 2021) pp. 181–210.

[40] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," (2017), https://doi.org/10.48550/arXiv.1706.02633, _eprint: 1706.02633.

[41] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," (2016), https://doi.org/10.48550/arXiv.1611.09904, _eprint: 1611.09904.

[42] L. Xu and G. Henkelman, "Adaptive kinetic monte carlo for first-principles accelerated dynamics," **129**, 114104 (2008).

[43] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," **378**, 686–707 (2019).