Electronic Supplementary Material (ESI) for Physical Chemistry Chemical Physics.
This journal is © the Owner Societies 2021

Supplementary Information

Deep Reinforcement Learning of Transition States

Jun Zhang[1,#], Yao-Kun Lei[2,#], Zhen Zhang[3], Xu Han[2], Maodong Li[1], Lijiang Yang[2], Yi Isaac Yang[1,*] and Yi Qin Gao[1,2,4,5,*]

[1] *Institute of Systems and Physical Biology, Shenzhen Bay Laboratory, 518055 Shenzhen, China*
[2] *Beijing National Laboratory for Molecular Sciences, College of Chemistry and Molecular Engineering, Peking University, 100871 Beijing, China.*
[3] *Department of Physics, Tangshan Normal University, 063000 Tangshan, China.*
[4] *Beijing Advanced Innovation Center for Genomics, Peking University, 100871 Beijing, China.*
[5] *Biomedical Pioneering Innovation Center, Peking University, 100871 Beijing, China.*

[#] These authors contributed equally to this work.
[*] Correspondence should be sent to yangyi@szbl.ac.cn (Y.I.Y) or gaoyq@pku.edu.cn (Y.Q.G).

TABLE OF CONTENTS

PART I. SUPPUPLEMTAL TEXTS

I. Value function in RL‡

The value function $0 \le p_w(\mathrm{TP}|\mathbf{R}) \le 1$ measures the probability of yielding a successful transition path from the initial configuration $\mathbf{R}$. $p_w(\mathrm{TP}|\mathbf{R})$ can be approximated by a neural network with a sigmoid output. However, this quantity is intrinsically random, hence cannot be simply fitted by regression. We proposed the following algorithm to optimize this function.

A. Training objective

Given an initial configuration leading to a successful shooting, $\mathbf{R}_i$, we sample N failed shooting points $\{\mathbf{R}_j\}_{j=1}^{N}$, and calculate the following contrastive loss as suggested by some graph learning and self-supervised learning studies, [1-2]

$$\mathcal{L}(w;\mathbf{R}_i) = -\log \frac{p_w(\mathbf{R}_i)}{p_w(\mathbf{R}_i) + \sum_{j=1}^{N} p_w(\mathbf{R}_j)} \tag{S1}$$

where $p_w(\mathbf{R})$ is a shorthand notation for $p_w(\mathrm{TP}|\mathbf{R})$.

Note that Eq. (S1) is similar to a soft-max function with a single output logit corresponding to $p_w(\mathbf{R}_i)$. Therefore, for any successful shooting $\mathbf{R}_i$, $\mathcal{L}(w;\mathbf{R}_i)$ is minimized if $p_w(\mathrm{TP}|\mathbf{R}_j)$ for any failed shooting $\mathbf{R}_j$ is negligible compared to $p_w(\mathrm{TP}|\mathbf{R}_i)$. In this sense, $\log p_w(\mathrm{TP}|\mathbf{R})$ takes a flavour of energy, and it is the difference of $\log p_w(\mathrm{TP}|\mathbf{R})$ but not the absolute value makes sense.

This loss function is sample-efficient, and it allows us to optimize p_w by shooting merely one or a few trajectories from a given configuration. In all the experiments, we exclusively shoot only one trajectory at a time for a given initial configuration.

B. Regularization

Since it is not the absolute value of $p_w(\mathrm{TP}|\mathbf{R})$ matters in Eq. (S1), to stabilize the training (particularly to avoid numerical instability in the case $p_w(\mathbf{R}) \ll 1$), we regularized Eq. (S1) with a cross-entropy (XE) term which helps limit the overall drift of $p_w(\mathrm{TP}|\mathbf{R})$,

$$\mathrm{XE}(w;\mathbf{R}_i) = -\log p_w(\mathbf{R}_i) \tag{S2}$$

Finally, we adopt mini-batch optimization for $\mathcal{L}(w;\mathbf{R}_i)$. For each optimization step, we randomly sample a small number of positive configurations, $\{\mathbf{R}_i\}_{i=1}^{m}$, and perform negative sampling for each of the positive sample. The mini-batch loss function thus reads,

$$\mathcal{L}(w) \approx \frac{1}{m}\sum_{i=1}^{m}\left[\mathcal{L}(w;\mathbf{R}_i) + \lambda \mathrm{XE}(w;\mathbf{R}_i)\right] \tag{S3}$$

The mini-batch size m and the regularization strength λ are two hyper-parameters. The size of the accompany negative samples, N, is also a hyper-parameter, and usually a value between 10 to 50 suffices as suggested by related studies. [1, 3]

C. Susceptibility

Given the value function $p_w(\mathrm{TP}|\mathbf{R})$, we can analyze the relevance of each atom's coordinates to the transition mechanism. Intuitively, if an atom plays a more important role in the chemical transition, then the perturbation of its position would cause more dramatic changes in $p_w(\mathrm{TP}|\mathbf{R})$. In light of this reasoning, we define the "susceptibility" (denoted by $S(\mathbf{r}_i)$) of the value function w.r.t. the atom's position as follows, ,

$$S(\mathbf{r}_i) = \left\|\nabla_{\mathbf{r}_i} p_w(\mathrm{TP}|\mathbf{R})\right\| \tag{S4}$$

where $\mathbf{r}_i$ is the Cartesian coordinates of the i-th atom, and $\|\cdot\|$ denotes the L2-norm of the vector.

II. Policy function in RL‡

The policy function $V_\theta(\mathbf{R})$ is equivalent to a bias potential presented in most enhanced sampling approaches.[4-6] We proposed and experimented with two different optimization strategies for V_θ in this paper.

A. Policy gradient optimization (PGO)

One approach to update the parametrized $V_\theta(\mathbf{R})$ is via the expectation-maximization algorithm, also known as the policy gradient[7] in the literature of reinforcement learning. We thus term this optimization strategy as Policy Gradient Optimization (PGO). In PGO, we directly treat the value function $p_w(\mathrm{TP}|\mathbf{R})$ as a "reward", and define an "advantage" function $A(\mathbf{R})$ based on the reward function. One common choice of $A(\mathbf{R})$ is,

$$A(\mathbf{R}) = \log p_w(\mathrm{TP}|\mathbf{R}) - \left\langle \log p_w(\mathrm{TP}|\mathbf{R})\right\rangle_{V_\theta(\mathbf{R})} \tag{S5}$$

Note that $\langle \log p_w(\mathrm{TP}|\mathbf{R})\rangle_{V_\theta(\mathbf{R})}$ serves as a baseline in order to reduce the variance of the gradient estimator. It has been shown that any alternative function which is independent of the state (namely, $\mathbf{R}$) can also be used as the baseline. Then the unbiased gradient estimator of $V_\theta(\mathbf{R})$ reads,

$$\nabla_\theta \mathcal{L}(\theta) = \left\langle A(\mathbf{R})\beta V_\theta(\mathbf{R})\right\rangle_{V_\theta(\mathbf{R})} \tag{S6}$$

where β is the inverse temperature, and $\langle\cdot\rangle_{V_\theta(\mathbf{R})}$ denotes the expectation value over the equilibrium Boltzmann distribution biased by V_θ.

We remark here that, the performance of PGO strongly relies on the advantage function which is generally non-trivial to define, and careful reward shaping is usually needed. One of the main challenges in reward shaping is to balance the exploration with exploitation so as to avoid mode dropping (i.e., being early trapped in sub-optimal solutions). In this regard, Eq. (S6) could be less effective for

complicated reactions (e.g., where multiple distinct transition states exist).

B. Variational targeted optimization (VTO)

Another learning strategy is inspired by targeted adversarial optimized sampling (TALOS)[8] and variationally enhanced sampling (VES).[9] Specifically, given the value function $p_w(\mathrm{TP}|\mathbf{R})$, we can formulate a target distribution $p_\mathrm{T}(\mathbf{R})$ where regions of higher value also admit higher probability density. For instance, arguably the most straightforward choice of the target is to equate $p_\mathrm{T}(\mathbf{R})$ with $p_w(\mathrm{TP}|\mathbf{R})$. More reasonable choices of p_T which can better trade-off exploration and exploitation will be introduced in the next section.

Once the target distribution is set, our goal is to train a policy function $V_\theta(\mathbf{R})$ which could induce a biased equilibrium distribution, denoted by $p_\theta \propto \exp[-\beta(U + V_\theta)]$, identical to the target distribution p_T. This goal can be achieved via minimizing a strict divergence between p_T and p_θ. For example, in this paper we minimized the Kullback-Leibler divergence, $D_\mathrm{KL}(p_\mathrm{T}||p_\theta)$, following the gradient:[10-11]

$$\nabla_\theta D_\mathrm{KL}\left(p_\mathrm{T} \parallel p_\theta\right) = \left\langle \beta \nabla_\theta V_\theta \right\rangle_{p_\mathrm{T}} - \left\langle \beta \nabla_\theta V_\theta \right\rangle_{p_\theta} \qquad \text{(S7)}$$

Besides, the gradient for another strict divergence, Wasserstein-1 distance, $D_\mathrm{W}(p_\mathrm{T}||p_\theta)$, has been separately derived in TALOS,[8] and can also be used for optimizing $V_\theta(\mathbf{R})$.

III. Improved training of RL‡

Like any reinforcement learning tasks, RL‡ has to deal with issues including partial sampling and moving distributions (or dynamic datasets). Therefore, we adopted several useful techniques widely adopted in reinforcement learning, to make the training progress of RL‡ more robust and efficient.

A. Experience replay

It is known that the optimality of a value function is independent of the policy function (or the sampling procedure). So in order to reduce the dependence of the convergence of the value function on the policy function, we prepare two buffer sets $\mathcal{B}$, one containing all previously successful shootings (called positive buffer) and the other containing all failed shootings (called negative buffer).

In one Expert Iteration, we not only shoot trajectories initialized at the newly sampled configurations, but also at some configurations randomly drawn from both buffer sets. Consider that in a first trial, a particular configuration **R** leads to a successful transition path and enters into the positive buffer. However, **R** may also fail and hence into the negative buffer. Note that we do not override the buffers, so configurations with higher chances of winning will appear more frequently in the positive buffer. In this way, the randomness of the shooting outcomes is implicitly taken into account. During each optimization step, we randomly select a mini-batch of samples from both positive and negative buffers and feed them into Eq. (S3) to calculate the loss for the value function.

Furthermore, since generating data by real samplers (e.g. MD engines) are usually expensive, we also implement experience replay to reduce the cost of generating data and maximize the sample efficiency. Our reasoning is that, the recently sampled data are statistically close to the latest samples, hence can be jointly used via proper re-weighting according to importance sampling. This idea is similar to some reinforcement learning algorithms including trusted-region policy optimization[12] and proximal policy optimization.[13] Let $V^{(t)}(\mathbf{R})$ and $Z^{(t)}$ denote the bias potential and the corresponding partition function at the training iteration t, respectively. To replay the experience, one needs to estimate the ratio of the partition functions $Z^{(t)}$ between different t's. This can be conveniently done using multi-ensemble re-weighting techniques such as weighted histogram analysis (WHAM)[14] and multistate Bennett acceptance ratio (MBAR).[15]

B. Multi-agent exploration

In order to obtain a robust estimate of ensemble averages, we borrow the idea of advantage actor critic reinforcement learning[16] that to launch multiple samplers (or agents in terms of reinforcement learning) synchronously, and split the data collected by all the samplers into mini-batches to train the value and policy functions. Such practice effectively encourages exploration, not only leading to faster convergence, but also avoiding the early trap in the mode-collapsed local minima. In the research of enhanced sampling, similar strategy was adopted in the multiple-walker metadynamics[17] which usually ensures faster convergence than vanilla metadynamics.

C. Balance exploration and exploitation

PGO strongly relies on the choice of advantage or reward function. It can be found that a reward as defined in Eq. (S5) will be ineffective for PGO (Eq. (S6) if no positive feedback has been observed. Therefore, exploration should be accounted for separately if necessary. For example, one can design a reward function proportional to the free energy of the state. Besides, many enhanced sampling techniques can be used for exploration.

In VTO, we can readily balance the exploration with the exploitation by explicitly re-writing the target distribution into two components, one for exploration while the other for exploitation. The target distribution reads as follows (the same as Eq. (3) in the main text),

$$\log p_\mathrm{T} = \alpha \log p_\mathrm{exploit} + (1-\alpha) \log p_\mathrm{explore} \qquad \text{(S8)}$$

Given a set of order parameters $\mathbf{s}(\mathbf{R})$ which are used to define the reactant and product, we can define a uniform distribution over **s** as p_explore. A well-tempered target distribution over **s** can also be adopted for the same purpose.[11] Even if there are no order parameters available, one can still write the explorative distribution explicitly as in

integrated tempering sampling[18-19] or implicitly as in temperature replica exchange methods.[5]

As explained in the main text, the exploitive component is based on the value function,

$$p_{\text{exploit}} = p_w(\text{TP}\,|\,\mathbf{R}) \tag{S9}$$

The hyper-parameter α in Eq. (S8) is used to trade exploration for exploitation. When α is close to zero, the sampling is dominant by exploration; When α approaches 1, the sampling is dominant by exploitation. We thus term α as the "exploitation factor".

Since the exploitive distribution (Eq. (S9)) is learned from scratch and gradually updated during RL‡, it would be very inaccurate in the beginning of training. Moreover, at the beginning of the training when positive feedbacks (i.e., successful shootings) are rare, so exploration is very important. Following these considerations, during training, we recruited a schedule to gradually tune α from 0 to an asymptotic threshold $\alpha_{\max} \leq 1$. We found that usually a threshold $\alpha_{\max}$ between 0.5 and 0.6 suffices for enhanced sampling of the transition states, meanwhile preserves adequate exploration (or coverage) over the rest of the configurational space.

IV. Parametrization of RL‡

Many functional forms are optional for the value function policy function V_θ. For example, it can be a linear expansion of certain basis functions where θ are the expansion coefficients;[9] or can be a non-linear neural network, where θ are the built-in parameters of artificial neural networks (ANNs).[20] On the other end, the value function p_w is exclusively approximated by ANNs in this paper.

A. Orthonormal polynomials or functions

If the input to the policy network is low-dimensional, we recommend orthonormal basis functions as V_θ, and the expansion coefficients are the learnable parameters. For periodic input variables $\mathbf{s}(\mathbf{R})$, like torsional angles, Fourier expansions can be adopted. For non-periodic $\mathbf{s}(\mathbf{R})$, Legendre or Chebyshev polynomials can be used. Orthonormal polynomials usually yield smooth energy function, so generally no additional regularization is needed.

B. Multi-layer perceptron (MLP)

MLPs are most commonly seen ANNs consisting of fully connected hidden layer. The input of a MLP, $\mathbf{s}(\mathbf{R})$, should trans-rotational invariant features of the molecular system. More importantly, each dimension of the input vector $\mathbf{s}$ should be indexible. MLP will transform the input vector to hidden features, and finally yields an output vector. We experimented with MLP in conjunction with neural allocative potential[11] as a policy function in the numerical model system.

C. Graph Neural Network (GNN)

Molecular systems consisting of particles (e.g., atoms) can be viewed as a graph, where vertices (or nodes) represent the particles while the interactions between particles can be modeled by graphical edges. Graph neural networks hence can be used to model molecular systems. GNNs exhibit a nice property that preserves the trans-rotational invariance and permutation invariance of the many-particle system.[21] There are several recently developed models that specifically deal with molecular systems, including SchNet[22] and PhysNet.[23] Both models directly learn a function based on the Cartesian coordinates $\mathbf{R}$ and the type of the particles. Particularly, in some experiments, SchNet was used as the value functions $p_w(\text{TP}|\mathbf{R})$ for post-analysis of the reaction mechanism.

D. Neural allocative potentials (NAP)

We note here that it is not the absolute value but the difference of energy makes physical sense. Based on this observation Zhang *et al.* proposed a new output layer for ANN-based potential energy functions, called neural allocative potentials (NAP)[11]. Here we recapitulated the main ideas behind NAP. Firstly, a lower bound and an upper bound for the bias potential is chosen, and "quantized" into K fixed levels $\{E_k\}_{k=1..K}$. Then the following functional form is introduced,

$$V_\theta(\mathbf{R}) = \sum_{k=1}^{K} \omega_k(\mathbf{R};\theta) E_k \tag{S10}$$

where $\sum_{k=1,K} \omega_k(\mathbf{s};\theta) = 1$ corresponds to the output of an ANN with a soft-max output layer. Therefore, NAP is nothing but a new output layer that can be straightforwardly equipped with any ANNs.

Now the problem of learning a scalar is transformed into learning a simplex $\{\alpha_k\}_{k=1..K}$. Following this form, NAP is trained to allocate proper amount of energy to configuration $\mathbf{s}$, rather than estimate the absolute value of the bias potential.

V. Expert Iterations in RL‡

Training of RL‡ is based on Expert Iterations (EXIT's).[24] In other words, RL‡ is gradually improved through a series of EXIT's. Each Expert Iteration consists of four individual steps that together form a loop:

Step 1. Sample according to p_{shoot} as Eq. (1) in the main text, where the Expert performs exploration over the configuration space.

Step 2. Shoot trajectories from the sampled coordinates with random momenta (Algorithm S1), and record the game results into replay buffers.

Step 3. Sample from the replay buffers and update the value function according to Eq. (S3).

Step 4. Update the policy function V_θ, either through PGO as Eq. (S6) or VTO as Eq. (S7). Use the updated V_θ for p_{shoot} (thus the Expert is improved), and return to Step 1.

We remark here that, Steps 1 and 2 correspond to the Expert's action, to discover promising configurations leading to the victory. Steps 3 and 4 correspond to the training of the Apprentice, who summarizes the positive

feedbacks provided by the Expert. Based on the improved Apprentice (i.e., the updated value and policy functions), the action of the Expert is optimized. Synapses of the "shooting game" and the training protocol of RL‡ are summarized in Algorithm S1 and Algorithm S2, respectively.

As in many reinforcement learning tasks, it is hard to define or reach a definitive optimal solution, so usually we will stop at a fairly satisfying model. We stop the training of RL‡ according to two criteria: i) The loss for value function, Eq. (S3), becomes steady and no longer diminishes, and ii) The chance of winning the game according to p_{shoot} exceeds a user-specified threshold or no longer increases (the default threshold in this paper is 10%). For ANNs, we adopt Adam[25] as the optimizer. If basis functions are used as the policy function, the Averaged Stochastic Gradient Descent (ASGD)[26] is optional as optimizer as suggested by VES.[9] Besides, by adopting the advanced optimization techniques including experience replay and multi-agent exploration as introduced earlier, we can further expedite and stabilize the training of RL‡.

Algorithm S1. Shooting game in RL‡

Define: reactant and product; inverse-temperature β; step-size Δt, maximal simulation length $t_{\max}$.

Input: initial configurations $\{\mathbf{R}_0\}$.

Sample random momenta $\{\mathbf{p}_0\}$ according to Maxwell distribution at β.

Initialize Leap-Frog integrator at $\{\mathbf{R}_0, \mathbf{p}_0\}$

While $\mathbf{R}_t$ not belong to reactant or product and $t < t_{\max}$, **do**

  $\mathbf{R}_t, \mathbf{p}_t \leftarrow \text{LeapFrog}\left(\mathbf{R}_{t-1}, \mathbf{p}_{t-1}, \Delta t\right)$ ▷ Forward integration

End While

While $\mathbf{R}_{-t}$ not belong to reactant or product, **do**

  $\mathbf{R}_{-t}, \mathbf{p}_{-t} \leftarrow \text{LeapFrog}\left(\mathbf{R}_{-(t-1)}, \mathbf{p}_{-(t-1)}, -\Delta t\right)$ ▷ Reversed integration

End While

If $\{\mathbf{R}_t\}$ and $\{\mathbf{R}_{-t}\}$ end up in different states, we win the game.

Algorithm S2. Reinforcement Learning of Transition States (RL‡)

Input: Initialize value network p_w , policy network V_θ and void replay buffers $\mathcal{B}$.
Set learning rates α_w and α_θ for w and θ, and optimizer's hyper-parameters
While converge criteria not met, **do**
  Run MD/MC under $U + V_\theta$, collect samples $\{\mathbf{R}_{\mathrm{MD}}\}$ ▷ Expert's action
  Draw random samples $\{\mathbf{R}_{\mathcal{B}}\}$ from replay buffers ▷ experience replay
  Shooting from $\{\mathbf{R}_{\mathrm{MD}}\} \cup \{\mathbf{R}_{\mathcal{B}}\}$ according to Algorithm S1.
  Update replay buffers $\mathcal{B}$
  For $t = 0 \to n_w$ ▷ train p_w for n_w steps
    Draw mini-batch of samples from $\mathcal{B}$ to calculate $\mathcal{L}(w)$. ▷ Eq. (S3)
    $w \leftarrow \mathrm{Adam}\left(\nabla_w \mathcal{L}(w), w, \alpha_w\right)$ ▷ update value network
  End For
  If VTO, **do**
    Update the exploitation factor α.
    Update the target distribution p_{T}. ▷ Eq. (S8)
  End If
  For $t = 0 \to n_\theta$ ▷ train V_θ for n_θ steps
    Draw mini-batch of samples from $\{\mathbf{R}_{\mathrm{MD}}\}$ to calculate $\langle \nabla_\theta V_\theta(\mathbf{s}) \rangle_{p_\theta}$.
    Calculate $\nabla_\theta \mathcal{L}(\theta)$ via PGO or VTO ▷ Eq. (S6) or (S7)
    $\theta \leftarrow \mathrm{Adam}\,/\,\mathrm{ASGD}\left(\nabla_\theta \mathcal{L}(\theta), \theta, \alpha_\theta\right)$ ▷ update policy network
  End For
End While

PART II. SIMULATION AND TRAINING DETAILS

I. Berezhkovskii-Szabo potential

We first illustrated how RL‡ works in an interpretable way on a model system, the Berezhkovskii-Szabo (BS) potential.[27] This 2-dimensional model potential consists of two local minima separated by an energy barrier (Fig. S1A), and captures key features of chemical reactions in a simplified manner.

A. Simulation setup

Given the inverse temperature β, the BS potential[27] takes the following form:

$$\beta U(x,y) = \beta U(x) + \frac{\Omega^2 (x-y)^2}{2}$$

where

$$\beta U(x) = -\Delta + \frac{\omega^2 (x+x_0)^2}{2}, x < -\frac{x_0}{2}$$

$$= -\frac{\omega^2 x^2}{2}, -\frac{x_0}{2} < x < \frac{x_0}{2}$$

$$= -\Delta + \frac{\omega^2 (x-x_0)^2}{2}, x > \frac{x_0}{2}$$

For our simulations, we chose $\beta = 1$, $x_0 = 2.2$, $\omega^2 = 4$, $\Delta = \omega^2 {x_0}^2/4$, and $\Omega^2 = 1.01\omega^2$. Besides, the diagonal diffusion tensor used in the overdamped Langevin equation was set to be:

$$\mathbf{D} = \begin{pmatrix} D_{xx} & 0 \\ 0 & D_{yy} \end{pmatrix} = \begin{pmatrix} \frac{2}{3\beta} & 0 \\ 0 & \frac{2}{3\beta} \end{pmatrix}$$

The resulting white-noised Langevin dynamics was simulated with a discrete time integration step of 0.01.

B. RL‡ models

In the shooting game, the reactant is defined as $(x < -2, y < -2)$, and the product is $(x > 2, y > -2)$.

The value function $p_w(\mathrm{TP}|(x,y))$ is a MLP which inputs (x,y). The MLP contains 3 hidden layers, each of 64 units and an ELU activation function.[28] The output layer corresponds to a single unit with sigmoid activation function.

The policy function $V_\theta(x,y)$ is also a MLP, which inputs (x,y) and contains 2 hidden layers with hyperbolic tangent activation. The output layer takes the form of NAP (Eq. (S10)), consisting of 11 logits which uniformly quantize the energy range of $[-5,5]$ k_BT. Besides, we implemented spectral normalization[29] over the weight matrices of the MLP to regularize the gradients.

C. Training details

We regularized the value network as in Eq. (S2), with the regularization strength $\lambda = 0.1$. We optimized the value network according to Eq. (S3), with a mini-batch $m = 10$ positive samples, and each positive sample is accompanied by $N = 50$ negative samples. The Adam optimizer with a learning rate of 10^{-3} was adopted (the hyper-parameter $\beta_1 = 0.8$ and $\beta_2 = 0.9$).

Given a policy function V_θ, we ran a biased Langevin dynamics for 5000 steps, and used the collected samples to update the policy network. In this model, such a short Langevin simulation suffices to generate near-equilibrium samples, so we did not perform additional exploration or pre-training, and directly optimized the policy function via PGO (Eq. (S6)). The Adam optimizer with default hyper-parameters and a learning rate of 2×10^{-4} was adopted.

In one Expert Iteration, $n_w = n_\theta = 5$ steps of optimization were performed for both p_w and V_θ, respectively (Algorithm S2). In each EXIT, we launched 100 random shootings. Totally 100 Expert Iterations were executed, and the overall shooting moves amounts to 10,000. The final successful shooting probability is about 12% (Fig. S1B).

D. Results

From Fig. S1B we can see that the training of RL‡ is very efficient, leading to a converged value function (with $\mathcal{L}(w)$ no longer diminishing significantly) within 50 EXIT's, and the average winning chance exceeds the prescribed threshold (10%) after 80 EXIT's. Besides, RL‡ is fairly sample-efficient in that only 10,000 random shootings were performed in total. After training is done, we plotted the policy function (Fig. S1C) and found it clearly diverts the sampling from the reactant or product regions to the TS region. This form of V_θ reminds one of the harmonic bias potential centered at the TS as adopted in umbrella sampling. However, unlike umbrella sampling, V_θ in RL‡ is learned without any knowledge of the reaction coordinate. On the other end, we can interpret the reaction mechanism based on the value function (Fig. S1D). As can be seen in Fig. S1D, the region of highest $p_w(\mathrm{TP}|x,y)$ values corresponds exactly to the dividing surface of the BS potential. Additionally, we can track the training process of RL‡ by plotting the samples from p_{shoot} at different training iterations (Fig. S1E). We find that the sampled configurations become increasingly concentrated around the TS region. Particularly after 100 EXIT's, the sampled configurations are located exactly around the TS, in agreement with the expected effect brought by the optimized policy V_θ. This example demonstrates that RL‡ can be trained *tabula rasa* hence allows us to reveal the reaction mechanisms with minimal *a priori* expertise or assumptions.

II. S_N2 reaction

A. Simulation setup

The MD simulations of the S_N2 reaction in vacuum were performed using the QM/MM module of SANDER program at AmberTools 17 software package.[30] All the atoms of the system are set as the QM part using the semi-empirical PM6 model.[31] No constraint is added on chemical bonds, and no cut-off is used for the non-bonding interactions. The simulations temperature was kept at 300 K, and the time step was 1 fs. During the simulation, harmonic restraining walls had been added on d_1 and d_2 (Fig. 2A in the main text) for distances larger than 6 Å to prevent the atoms moving too far away, which was utilized using the bias UPPER_WALL in the PLUMED2[32] plug-in library (KAPPA is 1000 kJ/mol).

B. RL‡ models

In the shooting game, we selected $\mathbf{s} = (d_1, d_2)$ as the order parameter for characterization of reactant and product (Fig. 2A in the main text). The reactant is defined as $(1\text{Å} < d_1 < 2\text{Å}, d_2 > 4\text{Å})$, and the product is $(d_1 > 4\text{Å}, 1\text{Å} < d_2 < 2\text{Å})$.

During EXIT, we constructed a value function $p_w(\text{TP}|\mathbf{s})$ operating on **s** but not the coordinates of the entire system. p_w is a MLP contains 2 hidden layers, each of 128 units and an ELU activation function. The MLP inputs (d_1, d_2), and the vector yielded by the last hidden layer is denoted by $\mathbf{z}_{12}$. We took account of that p_w should be symmetric w.r.t. the exchange of d_1 and d_2, so we also applied the same MLP but with the input transposed, (d_2, d_1), and yielded another hidden vector $\mathbf{z}_{21}$. We then summed over $\mathbf{z}_{12}$ and $\mathbf{z}_{21}$ before feeding forward to the final sigmoid output unit.

The policy function $V_\theta(\mathbf{s})$ was also built upon **s**, and was expanded by Legendre polynomials,

$$V_\theta(d_1, d_2) = \sum_{i,j=0}^{K} \theta_{ij} f_i(\bar{d}_1) f_j(\bar{d}_2)$$

where f_i denotes the *i*-th order Legendre polynomial with $K = 25$. $\bar{d}$ is a generalized sigmoid function[33] which squashes d into the range of [-1,1] supported by the Legendre polynomials.

C. Training details

We regularized the value network $p_w(\text{TP}|\mathbf{s})$ with the regularization strength $\lambda = 0.5$ (Eq. (S2)). The value network was optimized according to Eq. (S3) with a mini-batch $m = 20$ positive samples, and each positive sample is accompanied by $N = 50$ negative samples. The Adam optimizer with a learning rate of 10^{-4} and default hyper-parameters was adopted.

Given a policy function V_θ, we performed multi-agent exploration with 10 parallel QM/MM MD simulations according to Eq. (1) in the main text, and each simulation lasted for 2 ps. The collected samples were then used to update the policy network via PGO (Eq. (S6)). Since the reaction can rarely take place in an unmodified PES, we performed a pre-training for exploration prior to PGO, where a metadynamics simulation over **s** was launched. The samples collected during metadynamics was used to approximate the FES (Fig. 2B in the main text). We also implemented experience replay to expedite the training of the policy network. We adopted Adam optimizer with a learning rate of 0.1 and the hyper-parameters were chosen as $\beta_1 = 0.8$ and $\beta_2 = 0.9$. Optimization of both functions was executed on Tensorflow (v1.15.0).

In one Expert Iteration, $n_w = 10$ steps of optimization were performed for p_w and $n_\theta = 2$ for V_θ, respectively (Algorithm S2). In each EXIT, we launched 100 random shootings. Totally 300 Expert Iterations were executed, and the overall shooting moves amounts to 30,000. The final successful shooting probability is about 15%.

D. Susceptibility analysis

After EXIT, we trained a new value function $p_{w'}(\text{TP}|\mathbf{R})$ based on the obtained transition state ensembles. $p_{w'}(\text{TP}|\mathbf{R})$ operates on the Cartesian coordinates of all the atoms. $p_{w'}(\text{TP}|\mathbf{R})$ takes the default architecture of a SchNet,[22] except that we replaced the original the output layer by a single output unit with a sigmoid activation function.

Since the training will suffer from disappeared gradient during backpropagation due to sigmoid function, then we regularized the input to sigmoid layer $\mathbf{h}(\mathbf{R}_i)$ according to equation below with the regularization strength $\lambda_{\text{EC}} = 10^{-4}$.

$$\text{EC}(w'; \mathbf{R}_i) = \lambda_{\text{EC}} \left| \mathbf{h}(w'; \mathbf{R}_i) \right|^2 \qquad \text{(S11)}$$

We also regularized the value network $p_{w'}(\text{TP}|\mathbf{R})$ with the regularization strength $\lambda = 0.001$ (Eq. (S2)). The value network was optimized with a mini-batch $m = 10$ positive samples, and each positive sample is accompanied by $N = 20$ negative samples.

After optimization of $p_{w'}(\text{TP}|\mathbf{R})$ was done, the susceptibility was computed for each atom according to Eq. (S4). The susceptibility of each atom was then averaged over the transition state ensemble and reported in Fig. 2E in the main text.

III. Alanine dipeptide (Ala2)

A. Simulation setup

For the alanine dipeptide in aqueous solution, no ions were added since the terminal of the alanine was neutrally blocked (namely, ACE-ALA-NME) surrounded by 384 SPCE[34] water molecules. All the simulations were executed on AMBER17 package[30] using FF99SB force field [35] parameters. The aqueous solution system was put in a rectangular simulation box with periodic boundaries on. SHAKE algorithm[36] was adopted to constrain all covalent bonds involving hydrogen atoms, and a 2 fs time step was permitted. The system underwent a standard relaxation procedure and equilibrated to an NTP ensemble (300 K, 1atm). To equilibrate the system to the appropriate volume, the pressure of the system was adjusted to 1 atm by the Berendsen weak-coupling algorithm[37] with the relaxation time constants of 0.2 ps under another 1 ns long normal MD.

B. RL‡ models

In the shooting game, we defined the reactant, i.e., the *cis*-conformation, corresponding to $30° < \phi < 90°$; while the product, i.e., the *trans*-conformation, corresponds to $-170° < \phi < -60°$.

During EXIT training, we constructed the value on $\mathbf{s} = (\phi, \varphi)$. The value function $p_w(\mathrm{TP}|\mathbf{s})$ is a MLP. To account for the periodicity of $\mathbf{s}$, we first transform $\mathbf{s}$ into a torus vector,

$$\mathbf{s}_{\mathrm{torus}} = \left(\cos\phi, \cos\varphi, \sin\phi, \sin\varphi\right)$$

Then $\mathbf{s}_{\mathrm{torus}}$ is fed into the MLP composed of 2 hidden layers, each with 128 hidden units and ELU activation function. The output layer corresponds to a single unit with sigmoid activation function.

The policy $V_\theta(\mathbf{s})$ is also a function of the 2D (ϕ, φ) space. Since ϕ and φ are periodic variables, we expanded $V_\theta(\phi, \varphi)$ using the Fourier polynomial:

$$V_\theta = \sum_{i=0}^{i=N} \sum_{j=0}^{j=N} \theta_{ij} \left[\cos(i\phi) + \sin(i\phi)\right]\left[\cos(j\varphi) + \sin(j\varphi)\right]$$

where $N = 8$.

C. Training details

The value network $p_w(\mathrm{TP}|\mathbf{s})$ was regularized according to Eq. (S2) with $\lambda = 0.5$. We optimized the value network using a mini-batch of $m = 20$ positive samples, and each positive sample is accompanied by $N = 50$ negative samples. The Adam optimizer with a learning rate of 10^{-4} and default hyper-parameters was adopted. The optimization of $p_w(\mathrm{TP}|\mathbf{s})$ was done in Tensorflow (v1.15.0).

Under a policy function $V_\theta(\mathbf{s})$, we performed multi-agent exploration with 10 parallel MD simulations according to Eq. (1) in the main text, and each trajectory was 20 ps in length. Experience replay was also adopted to expedite the training of V_θ. The collected samples were used to update V_θ via VTO as in Eq. (S7). We chose a uniform distribution over $\mathbf{s}$ as p_{explore}, and the target distribution p_{T} was defined according to Eq. (S8). During training, the exploitation factor α was gradually increased from 0 to 0.6. We implemented ASGD optimizer built in PLUMED2, with a learning rate of 0.5 to update V_θ.

In one Expert Iteration, $n_w = 10$ steps of optimization were performed for p_w and $n_\theta = 200$ for V_θ, respectively (Algorithm S2). In each EXIT, we launched 100 random shootings. Totally 240 Expert Iterations were executed, and the overall shooting moves amounts to 24,000. The final successful shooting probability is about 11%.

D. Susceptibility analysis

After EXIT, we trained a new value function $p_{w'}(\mathrm{TP}|\mathbf{R})$ based on the obtained transition state ensembles. The model architecture is the same as in S_N2 reaction.

The value network $p_w(\mathrm{TP}|\mathbf{s})$ was regularized according to Eq. (S2) and Eq. (S11) with $\lambda = 0.01$ and $\lambda_{\mathrm{EC}} = 10^{-4}$. We optimized the value network using a mini-batch of $m = 10$ positive samples, and each positive sample is accompanied by $N = 20$ negative samples. The Adam optimizer with a learning rate of 10^{-4} and default hyper-parameters was adopted. The optimization of $p_{w'}(\mathrm{TP}|\mathbf{R})$ was done in Tensorflow (v1.15.0).

After optimization of $p_{w'}(\mathrm{TP}|\mathbf{R})$ was done, the susceptibility was computed for each atom according to Eq. (S4). Since we are interested in the solvent effects, we calculated the radial averaged susceptibility for the solvent atoms. Specifically, for a given solvent atom type z (e.g., oxygen or hydrogen), RAS of atom type z as a function of radial distance r is defined as,

$$\mathrm{RAS}(r; z) = \frac{\int S(\mathbf{r}_z)\,\delta\left(\|\mathbf{r}_z - \mathbf{R}_{\mathrm{COM}}\| - r\right) d\mathbf{r}_z}{\int \delta\left(\|\mathbf{r}_z - \mathbf{R}_{\mathrm{COM}}\| - r\right) d\mathbf{r}_z} \qquad \text{(S12)}$$

where $\boldsymbol{r}_z$ stands for the positions of all z-type atoms. $S(\boldsymbol{r}_z)$ is defined as Eq. (S4), $\mathbf{R}_{\mathrm{COM}}$ is the coordinate of Ala2's center-of-mass, and δ denotes the Dirac-delta function. We computed the RAS for oxygen and hydrogen, respectively, and reported the results in Fig. 3G in the main text.

IV. Claisen rearrangement

A. Simulation setup

The simulation was performed at the QM/MM interface on AMBER14 MD platform. The self-consistent charge density functional tight-binding (SCC-DFTB) method [38] was adopted to approximate the quantum mechanical Hamiltonian of the reactant molecule. The solvent is a kind of ionic liquid, containing a pair of soluble ion pairs termed as $[C_2mim]^+$ $[NTf_2]^-$ (Fig. S3). We adopted the classical force field developed by Sieffert and Wipff[39-40] to describe the solvent molecular ions and SHAKE was imposed on the solvent. No additional ions were added.

The system underwent a standard relaxation procedure and equilibrated to an NTP ensemble (300 K, 1 atm) lasting for 1-ns long normal MD. A cutoff of 10.0 Å was applied for calculating nonbonding interactions. All the simulations were performed with a 1-fs time integration step (no SHAKE on QM-treated molecule) and with periodic boundary condition.

B. RL‡ models

In the shooting game, we selected $\mathbf{s} = (d_1, d_2)$ as the order parameter to distinguish the reactant from product (Fig. 4A in the main text). The reactant is defined as $(1\text{Å} < d_1 < 1.6\text{Å}, d_2 > 2.5\text{Å})$, and the product is $(d_1 > 3\text{Å}, 1\text{Å} < d_2 < 1.5\text{Å})$.

During EXIT training, a MLP which inputs $\mathbf{s}$ was employed as the value function $p_w(\mathrm{TP}|\mathbf{s})$. This MLP is stacked with 2 hidden layers, each consisting of 128 units with ELU activation function. The output layer is a single sigmoid unit.

The policy function $V_\theta(\mathbf{s})$, which also operates on the 2D (d_1, d_2) space, is expanded by Legendre polynomials. The input d was first linearly scaled into $-1 \le \bar{d} \le 1$, and V_θ takes the form of

$$V_\theta(d_1, d_2) = \sum_{i,j=0}^{K} \theta_{ij} f_i(\bar{d}_1) f_j(\bar{d}_2)$$

where f_i denotes the i-th order Legendre polynomial, and $K = 30$.

C. Training details

We regularized the value network $p_w(\mathrm{TP}|\mathbf{s})$ according to Eq. (S2) with $\lambda = 0.5$, and minimized Eq. (S3) using a mini-batch of $m = 20$ positive samples. Each positive sample was accompanied by $N = 50$ negative samples. The Adam optimizer with a learning rate of 10^{-4} and default hyper-parameters was adopted. The optimization of $p_w(\mathrm{TP}|\mathbf{s})$ was carried out on Tensorflow (v1.15.0).

In each training iteration, 10 parallel QM/MM MD simulations were performed for multi-agent exploration under a biased potential $V_\theta(\mathbf{s})$ according to Eq. (1) in the main text, and each individual simulation was 20-ps long. The collected samples were used to update V_θ via VTO as in Eq. (S7). We chose a uniform distribution over $\mathbf{s}$ as p_{explore}, and the target distribution p_{T} was defined according to Eq. (S8). During training, the exploitation factor α was gradually increased from 0 to 0.5. We implemented ASGD optimizer built in PLUMED2, with a learning rate of 0.5 to update V_θ.

In one Expert Iteration, $n_w = 10$ steps of optimization were performed for p_w, and $n_\theta = 100$ for V_θ, respectively (Algorithm S2), and 100 random shootings were launched. Totally 400 Expert Iterations were executed, and the overall shooting moves amounts to 40,000. The final successful shooting probability is about 10%.

D. Susceptibility analysis

After EXIT, we trained a new value function $p_{w'}(\mathrm{TP}|\mathbf{R})$ based on the obtained transition state ensembles. The model architecture is the same as in S_N2 reaction. Since the solvent contains too many atoms, we first coarse-grained each solvent molecule into a single particle located at its center-of-mass. $p_{w'}(\mathrm{TP}|\mathbf{R})$ then incorporates the coordinates of all the solute atoms and the coarse-grained solvent particles.

The value network $p_{w'}(\mathrm{TP}|\mathbf{R})$ was regularized according to Eq. (S2) and Eq. (S11) with $\lambda = 0.01$ and $\lambda_{\mathrm{EC}} = 10^{-4}$. We optimized the value network using a mini-batch of $m = 10$ positive samples, and each positive sample is accompanied by $N = 20$ negative samples. The Adam optimizer with a learning rate of 10^{-4} and default hyper-parameters was adopted. The optimization was done in Tensorflow (v1.15.0).After optimization of $p_{w'}(\mathrm{TP}|\mathbf{R})$ was done, the susceptibility was computed for each solute atom according to Eq. (S4), and was then averaged over the transition state ensemble and reported in Fig. 4E. We also calculated the radial averaged susceptibility (RAS) for the solvent molecules (as shown in Fig. 4F). Besides, we showcased the instantaneous susceptibility (i.e., not averaged over any other structures) of a snapshot (Fig. 4G), where each solvent molecule was colored as a whole according to the susceptibility of its corresponding coarse-grained particle.

PART III. SUPPLEMENTAL FIGURES

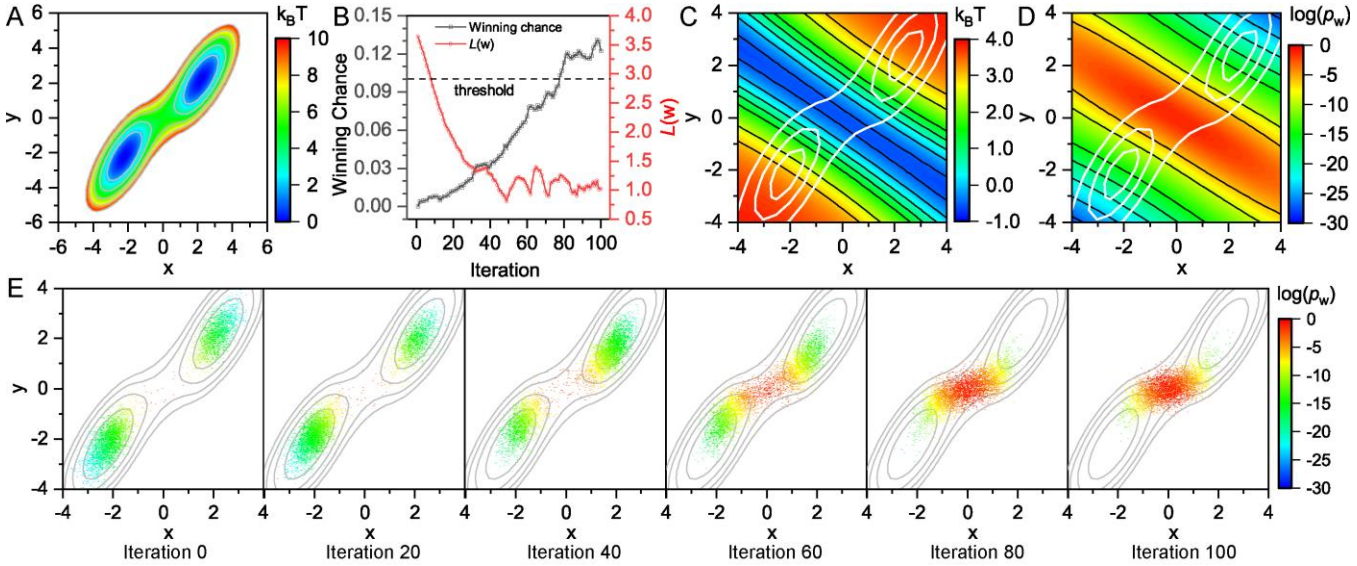


Figure S1. RL‡ for BS model potential. (A) Colored contour map of the BS potential energy surface (PES). (B) Average winning chance (black line) and loss of the value function $\mathcal{L}(w)$ (red line) plotted against Expert Iterations. A typical stop-threshold for training (winning chance exceeds 10%) is shown in dashed line. Indeed, we trained the value and policy functions for 100 iterations. (C) Contour map of the final policy function $V_\theta(x, y)$. Transparent contour lines of the PES are shown in background. (D) Contour map of the final value function $\log p_w(\mathrm{TP}|x, y)$. Transparent contour lines of the PES are shown in background. (E) Samples from p_{shoot} at different Expert Iterations, colored according to the value function $\log p_w(x, y)$; Grey contour lines of the PES are shown in background.

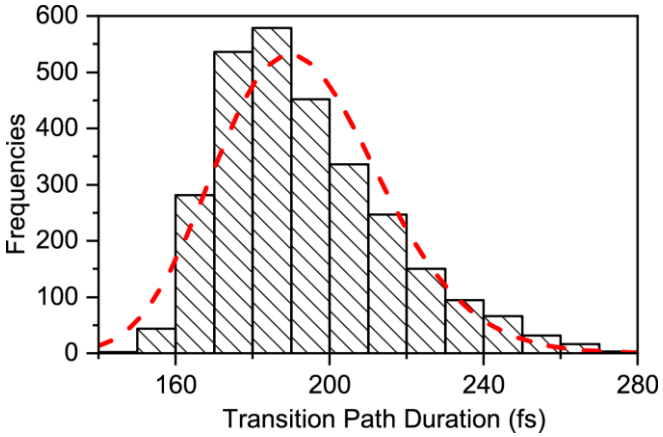
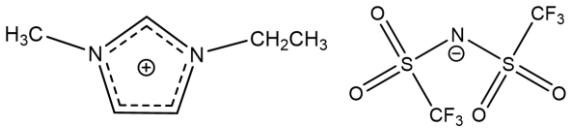


Figure S2. Statistics of the transition path duration (TPD) of the S_N2 reaction. The histogram of the TPD was computed according to the transition path ensemble collected by RL‡. The red dashed line stands for a log-normal fit of the TPD histogram.



[C_2mim]+ [NTf_2]-

Figure S3. Structures of the ionic liquid used as solvent for the Claisen rearrangement. The cation is [C_2min]+ (left) and the anion is [NTf_2]- (right).

REFERENCES

1. Hamilton, W.; Ying, Z.; Leskovec, J. In *Inductive representation learning on large graphs*, Advances in Neural Information Processing Systems, 2017; pp 1024-1034.
2. Oord, A. v. d.; Li, Y.; Vinyals, O., Representation Learning with Contrastive Predictive Coding. 2018.
3. Nickel, M.; Kiela, D. In *Poincar éembeddings for learning hierarchical representations*, Advances in neural information processing systems, 2017; pp 6338-6347.
4. Abrams, C.; Bussi, G., Enhanced sampling in molecular dynamics using metadynamics, replica-exchange, and temperature-acceleration. *Entropy* **2014,** *16* (1), 163-199.
5. Okamoto, Y., Generalized-ensemble algorithms: enhanced sampling techniques for Monte Carlo and molecular dynamics simulations. *Journal of Molecular Graphics and Modelling* **2004,** *22* (5), 425-439.
6. Yang, Y. I.; Shao, Q.; Zhang, J.; Yang, L.; Gao, Y. Q., Enhanced sampling in molecular dynamics. *The Journal of Chemical Physics* **2019,** *151* (7), 070902.
7. Williams, R. J., Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* **1992,** *8* (3-4), 229-256.
8. Zhang, J.; Yang, Y. I.; Noé, F., Targeted Adversarial Learning Optimized Sampling. *journal of physical chemistry letters* **2019,** *10* (19), 5791-5797.
9. Valsson, O.; Parrinello, M., Variational approach to enhanced sampling and free energy calculations. *Physical review letters* **2014,** *113* (9), 090601.
10. Chaimovich, A.; Shell, M. S., Coarse-graining errors and numerical optimization using a relative entropy framework. *The Journal of chemical physics* **2011,** *134* (9), 094112.
11. Zhang, J.; Lei, Y.; Yang, Y. I.; Gao, Y., Deep Learning for Variational Multi-Scale Molecular Modeling. *ChemRxiv. Preprint.* **2019,** *https://doi.org/10.26434/chemrxiv.9640814.v4*.
12. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P., Trust Region Policy Optimization. In *International Conference on Machine Learning*, 2015; pp 1889-1897.
13. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O., Proximal Policy Optimization Algorithms. 2017.
14. Kumar, S.; Rosenberg, J. M.; Bouzida, D.; Swendsen, R. H.; Kollman, P. A., The weighted histogram analysis method for free - energy calculations on biomolecules. I. The method. *Journal of computational chemistry* **1992,** *13* (8), 1011-1021.
15. Shirts, M. R.; Chodera, J. D., Statistically optimal analysis of samples from multiple equilibrium states. *The Journal of chemical physics* **2008,** *129* (12), 124105.
16. Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. In *Asynchronous methods for deep reinforcement learning*, International conference on machine learning, 2016; pp 1928-1937.
17. Raiteri, P.; Laio, A.; Gervasio, F. L.; Micheletti, C.; Parrinello, M., Efficient reconstruction of complex free energy landscapes by multiple walkers metadynamics. *The journal of physical chemistry B* **2006,** *110* (8), 3533-3539.
18. Gao, Y. Q., An integrate-over-temperature approach for enhanced sampling. *The Journal of chemical physics* **2008,** *128* (6), 064105.
19. Yang, L.; Liu, C.-W.; Shao, Q.; Zhang, J.; Gao, Y. Q., From thermodynamics to kinetics: enhanced sampling of rare events. *Accounts of chemical research* **2015,** *48* (4), 947-955.
20. Schneider, E.; Dai, L.; Topper, R. Q.; Drechsel-Grau, C.; Tuckerman, M. E., Stochastic neural network approach for learning high-dimensional free energy surfaces. *Physical review letters* **2017,** *119* (15), 150601.
21. Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, H. F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M.; Vinyals, O.; Li, Y.; Pascanu, R., Relational inductive biases, deep learning, and graph networks. 2018.
22. Schütt, K. T.; Kindermans, P. J.; Sauceda, H. E.; Chmiela, S.; Tkatchenko, A.; Müller, K. R., SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Neural Information Processing Systems*, 2017; pp 992-1002.
23. Unke, O. T.; Meuwly, M., PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *Journal of Chemical Theory and Computation* **2019,** *15* (6), 3678-3693.
24. Anthony, T.; Tian, Z.; Barber, D. In *Thinking fast and slow with deep learning and tree search*, Advances in Neural Information Processing Systems, 2017; pp 5360-5370.
25. Kingma, D. P.; Ba, J., Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**.
26. Bach, F.; Moulines, E., Non-strongly-convex smooth stochastic approximation with convergence rate O(1/n). In *Neural Information Processing Systems*, 2013; pp 773-781.
27. Berezhkovskii, A.; Szabo, A., One-dimensional reaction coordinates for diffusive activated rate processes in many dimensions. *The Journal of chemical physics* **2005,** *122* (1), 014503.

28. Clevert, D.-A.; Unterthiner, T.; Hochreiter, S., Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *International Conference on Learning Representations*, 2016.
29. Miyato, T.; Kataoka, T.; Koyama, M.; Yoshida, Y., Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* **2018**.
30. Case, D.; Cerutti, D.; Cheatham III, T.; Darden, T.; Duke, R.; Giese, T.; Gohlke, H.; Goetz, A.; Greene, D.; Homeyer, N., AMBER 2017, 2017. *San Francisco: University of California*.
31. Stewart, J. J., Optimization of parameters for semiempirical methods V: modification of NDDO approximations and application to 70 elements. *Journal of Molecular modeling* **2007,** *13* (12), 1173-1213.
32. Tribello, G. A.; Bonomi, M.; Branduardi, D.; Camilloni, C.; Bussi, G., PLUMED 2: New feathers for an old bird. *Computer Physics Communications* **2014,** *185* (2), 604-613.
33. Ceriotti, M.; Tribello, G. A.; Parrinello, M., Simplifying the representation of complex free-energy landscapes using sketch-map. *Proceedings of the National Academy of Sciences* **2011,** *108* (32), 13023-13028.
34. Berendsen, H.; Grigera, J.; Straatsma, T., The missing term in effective pair potentials. *Journal of Physical Chemistry* **1987,** *91* (24), 6269-6271.
35. Hornak, V.; Abel, R.; Okur, A.; Strockbine, B.; Roitberg, A.; Simmerling, C., Comparison of multiple Amber force fields and development of improved protein backbone parameters. *Proteins: Structure, Function, and Bioinformatics* **2006,** *65* (3), 712-725.
36. Ryckaert, J.-P.; Ciccotti, G.; Berendsen, H. J. C., Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics* **1977,** *23* (3), 327-341.
37. Berendsen, H. J. C.; Postma, J. P. M.; Gunsteren, W. F. v.; DiNola, A.; Haak, J. R., Molecular dynamics with coupling to an external bath. *The Journal of Chemical Physics* **1984,** *81* (8), 3684-3690.
38. Elstner, M.; Porezag, D.; Jungnickel, G.; Elsner, J.; Haugk, M.; Frauenheim, T.; Suhai, S.; Seifert, G., Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties. *Phys. Rev. B* **1998,** *58*, 7260-7268.
39. Sieffert, N.; Wipff, G., The [BMI][Tf2N] Ionic Liquid/Water Binary System: A Molecular Dynamics Study of Phase Separation and of the Liquid−Liquid Interface. *The Journal of Physical Chemistry B* **2006,** *110* (26), 13076-13085.
40. Fu, J.; Yang, Y. I.; Zhang, J.; Chen, Q.; Shen, X.; Gao, Y. Q., Structural Characteristics of Homogeneous Hydrophobic Ionic Liquid–HNO3–H2O Ternary System: Experimental Studies and Molecular Dynamics Simulations. *The Journal of Physical Chemistry B* **2016,** *120* (23), 5194-5202.