nature COMMUNICATIONS

ARTICLE

https://doi.org/10.1038/s41467-020-18959-8 OPEN

Learning molecular dynamics with simple language model built upon long short-term memory neural network

Sun-Ting Tsai[1], En-Jui Kuo[2] & Pratyush Tiwary[3]✉

Recurrent neural networks have led to breakthroughs in natural language processing and speech recognition. Here we show that recurrent networks, specifically long short-term memory networks can also capture the temporal evolution of chemical/biophysical trajectories. Our character-level language model learns a probabilistic model of 1-dimensional stochastic trajectories generated from higher-dimensional dynamics. The model captures Boltzmann statistics and also reproduces kinetics across a spectrum of timescales. We demonstrate how training the long short-term memory network is equivalent to learning a path entropy, and that its embedding layer, instead of representing contextual meaning of characters, here exhibits a nontrivial connectivity between different metastable states in the underlying physical system. We demonstrate our model's reliability through different benchmark systems and a force spectroscopy trajectory for multi-state riboswitch. We anticipate that our work represents a stepping stone in the understanding and use of recurrent neural networks for understanding the dynamics of complex stochastic molecular systems.

[1] Department of Physics and Institute for Physical Science and Technology, University of Maryland, College Park, MD 20742, USA. [2] Department of Physics and Joint Quantum Institute, University of Maryland, College Park, MD 20742, USA. [3] Department of Chemistry and Biochemistry and Institute for Physical Science and Technology, University of Maryland, College Park, MD 20742, USA. ✉email: ptiwary@umd.edu

Recurrent neural networks (RNN) are a machine learning/artificial intelligence (AI) technique developed for modeling temporal sequences, with demonstrated successes including but not limited to modeling human languages[1–7]. A specific and extremely popular instance of RNNs are long short-term memory (LSTM)[8] neural networks, which possess more flexibility and can be used for challenging tasks such as language modeling, machine translation, and weather forecasting[6,9,10]. LSTMs were developed to alleviate the limitation of previously existing RNN architectures wherein they could not learn information originating from far past in time. This is known as the vanishing gradient problem, a term that captures how the gradient or force experienced by the RNN parameters vanishes as a function of how long ago did the change happen in the underlying data[11,12]. LSTMs deal with this problem by controlling flows of gradients through a so-called gating mechanism where the gates can open or close determined by their values learned for each input. The gradients can now be preserved for longer sequences by deliberately gating out some of the effects. This way it has been shown that LSTMs can accumulate information for a long period of time by allowing the network to dynamically learn to forget aspects of information. Very recently LSTMs have also been shown to have the potential to mimic trajectories produced by experiments or simulations[13], making accurate predictions about a short time into the future, given access to a large amount of data in the past. Similarly, another RNN variant named reservoir computing[14] has been recently applied to learn and predict chaotic systems[15]. Such a capability is already useful for instance in weather forecasting, where one needs extremely accurate predictions valid for a short period of time. In this work, we consider an alternate and arguably novel use of RNNs, specifically LSTMs, in making predictions that in contrast to previous work[13,15], are valid for very long periods of time but only in a statistical sense. Unlike domains such as weather forecasting or speech recognition where LSTMs have allowed very accurate predictions albeit valid only for short duration of time, here we are interested in problems from chemical and biological physics, where the emphasis is more on making statistically valid predictions valid for extremely long duration of time. This is typified for example through the use of the ubiquitous notion of rate constant for activated barrier crossing, where short-time movements are typically treated as noise, and are not of interest for being captured through a dynamical model.

Here we suggest an alternative way to use LSTM-based language model to learn a probabilistic model from the time sequence along some low-dimensional order parameters produced by computer simulations or experiments of a high-dimensional system. We also show by our computer simulations of different model systems that the language model can produce the correct Boltzmann statistics (as can other AI methods such as refs. [16,17]) but also the kinetics over a large spectrum of modes characterizing the dynamics in the underlying data. We highlight here a unique aspect of this calculation that the order parameter our framework needs could be arbitrarily far from the true underlying slow mode, often called reaction coordinate. This in turn dictates how long of a memory kernel must be captured which is in general a very hard problem to solve[18,19]. Our framework is agnostic to proximity from the true reaction coordinate and reconstructs statistically accurate dynamics in a wide range of order parameters. We also show how the minimization of loss function leads to learning the path entropy of a physical system, and establish a connection between the embedding layer and transition probability. Followed by this connection, we also show how we can define a transition probability through embedding vectors. We provide tests for Boltzmann statistics and kinetics for Langevin dynamics of model potentials, MD simulation of alanine dipeptide, and trajectory from single molecule force spectroscopy experiment on a multi-state riboswitch[20], respectively. We also compare our protocol with alternate approaches including Hidden Markov Models. Our work thus represents a new usage of a popular AI framework to perform dynamical reconstruction in a domain of potentially high fundamental and practical relevance, including materials and drug design.

Results

Molecular dynamics can be mapped into a sequence of characters. Our central rationale in this work is that molecular dynamics (MD) trajectories, adequately discretized in space and time, can be mapped into a sequence of characters in some languages. By using a character-level language model that is effective in predicting future characters given the characters so far in a sequence, we can learn the evolution of the MD trajectory that was mapped into the characters. The model we use is stochastic since it learns each character through the probability they appear in a corpus used for training. This language model consists of three sequential parts shown schematically in Fig. 1. First, there is an embedding layer mapping one-hot vectors to dense vectors, followed by an LSTM layer which connects input states and hidden states at different time steps through a trainable recursive function, and finally a dense layer to transform the output of LSTM to the categorical probability vector.

Specifically, here we consider as input a one-dimensional time series produced by a physical system, for instance through Langevin dynamics being undergone by a complex molecular system. The time series consist of data points $\{\xi^{(t)}\}$, where t labels the time step and $\xi \in \mathbb{R}$ is some one-dimensional collective variable or order parameter for the high-dimensional molecular system. In line with standard practice for probabilistic models, we

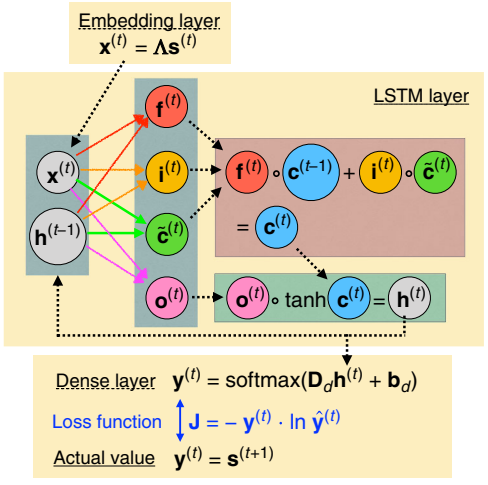

Fig. 1 Neural network schematic. The schematic plot of the simple character-level language model used in this work. The model consists of three main parts: The embedding layer, the LSTM layer, and a dense output layer. The embedding layer is a linear layer which multiplies the one-hot input $\mathbf{s}^{(t)}$ by a matrix and produces an embedding vector $\mathbf{x}^{(t)}$. The $\mathbf{x}^{(t)}$ is then used as the input of LSTM network, in which the forget gate $\mathbf{f}^{(t)}$, the input gate $\mathbf{i}^{(t)}$, the output gate $\mathbf{o}^{(t)}$, and the candidate value $\tilde{\mathbf{c}}^{(t)}$ are all controlled by $(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)})$. The forget gate and input gate are then used to produce the update equation of cell state $\mathbf{c}^{(t)}$. The output gate decides how much information propagates to the next time step. The output layer predicts the probabilities $\hat{\mathbf{y}}^{(t)}$ by parametrizing the transformation from $\mathbf{h}^{(t)}$ to $\hat{\mathbf{y}}$ with learned weights $\mathbf{D}_d$ and learned biases $\mathbf{b}_d$. Finally, we can compute the cross entropy between the predicted probability distribution $\hat{\mathbf{y}}^{(t)}$ and the true probability distribution $\mathbf{y}^{(t)} = \mathbf{s}^{(t+1)}$.

convert the data points to one-hot encoded representations that implement spatial discretization. Thus each data point $\{\xi^{(t)}\}$ is represented by a N-dimensional binary vector $\mathbf{s}^{(t)}$, where N is the number of discrete grid-points. An entry of one stands for the representative value and all the other entries are set to zeros. The representative values are in general finite if the order parameter is bounded, and are equally spaced in $\mathbb{R}$ with in total N representative values. Note that the time series $\{\xi^{(t)}\}$ does not have to be one-dimensional. For a higher-dimensional series, we can always choose a set of representative values corresponding to locations in the higher-dimensional space visited trajectory. This would typically lead to a larger N in the one-hot encoded representations, but the training set size itself will naturally stay the same. We find that the computational effort only depends on the size of training set and very weakly on N, and thus the time spent for learning a higher dimensional time series does not increase much relative to a one-dimensional series.

In the sense of modeling languages, the one-hot representation on its own cannot capture the relation between different characters. Take for instance that there is no word in the English language where the character c is followed by x, unless of course one allows for the possibility of a space or some other letter in between. To deal with this, computational linguists make use of an embedding layer. The embedding layer works as a look-up table which converts each one-hot vector $\mathbf{s}^{(t)}$ to a dense vector $\mathbf{x}^{(t)} \in \mathbb{R}^M$ by the multiplication of a matrix $\mathbf{\Lambda}$ which is called the embedding matrix, where M is called the embedding dimension

$$\mathbf{x}^{(t)} = \Lambda \mathbf{s}^{(t)} \tag{1}$$

The sequence of dense representation $\mathbf{x}^{(t)}$ accounts for the relation between different characters as seen in the training time series. $\mathbf{x}^{(t)}$ is then used as the input of the LSTM layer. Each $\mathbf{x}^{(t)}$ generates an output $\mathbf{h}^{(t)} \in \mathbb{R}^L$ from LSTM layer, where L is a tunable hyperparameter. Larger L generally gives better learning capability but needs more computational resources. The LSTM itself consists of the following elements: the input gate $\mathbf{i}^{(t)}$, the forget gate $\mathbf{f}^{(t)}$, the output gate $\mathbf{o}^{(t)}$ the cell state $\mathbf{c}^{(t)}$, the candidate value $\tilde{\mathbf{c}}^{(t)}$, and $\mathbf{h}^{(t)}$ which is the hidden state vector and the final output from the LSTM. Each gate processes information in different aspects.[8] Briefly, the input gate decides which information to be written, the forget gate decides which information to be erased, and the output gate decides which information to be read from the cell state to the hidden state. The update equation of these elements can be written as follows:

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{x}^{(t)} + \mathbf{U}_f \mathbf{h}^{(t-1)} + \mathbf{b}_f) \tag{2}$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{x}^{(t)} + \mathbf{U}_i \mathbf{h}^{(t-1)} + \mathbf{b}_i) \tag{3}$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_o \mathbf{x}^{(t)} + \mathbf{U}_o \mathbf{h}^{(t-1)} + \mathbf{b}_o) \tag{4}$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{x}^{(t)} + \mathbf{U}_c \mathbf{h}^{(t-1)} + \mathbf{b}_c) \tag{5}$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)} \tag{6}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh(\mathbf{c}^{(t)}) \tag{7}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the corresponding weight matrices and bias vectors. The $\tanh(\mathbf{v})$ operates piecewise on each element of the vector $\mathbf{v}$. The operation $\circ$ is the Hadamard product[21].

The final layer in Fig. 1 is a simple dense layer with fully connected neurons which converts the output $\mathbf{h}^{(t)}$ of the LSTM to a vector $\mathbf{y}^{(t)}$ in which each entry denotes the categorical probability of the representative value for the next time step $t+1$. The loss function J for minimization during training at every timestep t is then defined as the cross entropy between the output of the model $\hat{\mathbf{y}}^{(t)}$ and the actual probability for the next timestep $\mathbf{y}^{(t)}$ which is just the one-hot vector $\mathbf{s}^{t+1}$

$$\hat{\mathbf{y}}^{(t)} = \mathrm{softmax}(\mathbf{D}_d \mathbf{h}^{(t)} + \mathbf{b}_d) \tag{8}$$

$$J = -\sum_{t=0}^{T-1} \mathbf{y}^{(t)} \cdot \ln \hat{\mathbf{y}}^{(t)} = -\sum_{t=0}^{T-1} \mathbf{s}^{(t+1)} \cdot \ln \hat{\mathbf{y}}^{(t)} \tag{9}$$

where T is the total length of trajectory, and the final loss function is the sum over the whole time series. The $\mathrm{softmax}(\mathbf{x})_i = \exp(\mathbf{x}_i)/\sum_j \exp(\mathbf{x}_j)$ is a softmax function mapping $\mathbf{x}$ to a probability vector $\hat{\mathbf{y}}$.

Training the network is equivalent to learning path entropy. The central finding of this work, which we demonstrate through numerical results for different systems, is that a LSTM framework used to model languages can also be used to capture kinetic and thermodynamic aspects of dynamical trajectories prevalent in chemical and biological physics. In this section we demonstrate theoretically as to why LSTMs possess such a capability. Before we get into the mathematical reasoning detailed here, as well as in Supplementary Note 1, we first state our key idea. Minimizing the loss function J in LSTM (Eq. (9)), which trains the model at time t to generate output $\hat{\mathbf{y}}^{(t)}$ resembling the target output $\mathbf{s}^{t+1}$, is equivalent to minimizing the difference between the actual and LSTM-learned path probabilities. This difference between path probabilities can be calculated as a cross-entropy J' defined as:

$$J' = -\sum_{\mathbf{x}^{(T)}\ldots\mathbf{x}^{(0)}} P(\mathbf{x}^{(T)}\ldots\mathbf{x}^{(0)}) \ln Q(\mathbf{x}^{(T)}\ldots\mathbf{x}^{(0)}) \tag{10}$$

where $P(\mathbf{x}^{(t+1)}, \ldots, \mathbf{x}^{(0)})$ and $Q(\mathbf{x}^{(t+1)}, \ldots, \mathbf{x}^{(0)})$ are the corresponding true and neural network learned path probabilities of the system. Equation (10) can be rewritten[22] as the sum of path entropy $H(P)$ for the true distribution P and Kullback–Liebler distance D_{KL} between P and Q: $J' = H(P) + D_{KL}(P||Q)$. Since D_{KL} is strictly non-negative[22] attaining the value of 0 iff $Q = P$, the global minimum of J' happens when $Q = P$ and J' equals the path entropy $H(P)$ of the system.[23] Thus we claim that minimizing the loss function in LSTM is equivalent to learning the path entropy of the underlying physical model, which is what makes it capable of capturing kinetic information of the dynamical trajectory.

To prove this claim we start with rewriting J in Eq. (9). For a long enough observation period T or for a very large number of trajectories, J can be expressed as the cross entropy between conditional probabilities:

$$J = -\sum_{t=0}^{T-1} \sum_{\mathbf{x}^{(t+1)}} P(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}\ldots\mathbf{x}^{(0)}) \times \ln Q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}\ldots\mathbf{x}^{(0)}) \tag{11}$$

where $P(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}\ldots\mathbf{x}^{(0)})$ is the true conditional probability for the physical system, and $Q(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}\ldots\mathbf{x}^{(0)})$ is the conditional probability learned by the neural network. The minimization of Eq. (11) leads to minimization of the cross entropy J' as shown in the SI. Here we conversely show how Eq. (10) reduces to Eq. (9) by assuming a stationary first-order Markov process as in ref. [23]:

$$\begin{aligned} P(\mathbf{x}^{(T)}\ldots\mathbf{x}^{(0)}) &= P(\mathbf{x}^{(T)}|\mathbf{x}^{(T-1)})\ldots P(\mathbf{x}^{(1)}|\mathbf{x}^{(0)})P(\mathbf{x}^{(0)}) \\ Q(\mathbf{x}^{(T)}\ldots\mathbf{x}^{(0)}) &= Q(\mathbf{x}^{(T)}|\mathbf{x}^{(T-1)})\ldots Q(\mathbf{x}^{(1)}|\mathbf{x}^{(0)})Q(\mathbf{x}^{(0)}) \end{aligned} \tag{12}$$

where $P(\mathbf{x}_j^{(t+1)}|\mathbf{x}_i^{(t)}) \equiv P_{ij}$ is the transition probability from state $\mathbf{x}_i$ to state $\mathbf{x}_j$ and $P(\mathbf{x}_k^{(0)}) \equiv P_k$ is the occupation probability for the single state $\mathbf{x}_k$. Plugging Eq. (12) into Eq. (10), and following the

derivation in ref. [23] with the constraints

$$\sum_j P_{ij} = 1 \quad \sum_i P_i P_{ij} = P_j \tag{13}$$

we arrive at an expression for the cross-entropy J, which is very similar to the path entropy type expressions derived for instance in the framework of Maximum Caliber[23]:

$$J' = -\sum_i P_i \ln Q_i - T\sum_{lm} P_l P_{lm} \ln\left(Q_{lm}\right) \tag{14}$$

$$\rightarrow -T\sum_{lm} P(\mathbf{x}_l) P(\mathbf{x}_m|\mathbf{x}_l) \ln Q(\mathbf{x}_m|\mathbf{x}_l) \tag{15}$$

In Eq. (14) as the trajectory length T increases, the second term dominates in the estimate of J leading to Eq. (15). This second term is the ensemble average of a time-dependent quantity $\tilde{J}(\mathbf{x}_l^{(t)}) \equiv -\sum_m P(\mathbf{x}_m^{(t+1)}|\mathbf{x}_l^{(t)}) \ln Q(\mathbf{x}_m^{(t+1)}|\mathbf{x}_l^{(t)})$. For a large enough T, the ensemble average can be replaced by the time average. By assuming ergodicity[24]:

$$J' = -\sum_{t=1}^{T}\sum_m P(\mathbf{x}_m^{(t+1)}|\mathbf{x}_l^{(t)}) \ln Q(\mathbf{x}_m^{(t+1)}|\mathbf{x}_l^{(t)}) \tag{16}$$

from which we directly obtain Eq. (9). Therefore, under first-order Markovianity and ergodicity, minimizing the loss function J of Eq. (9) is equivalent to minimizing J' and thereby learning the path entropy. In the SI we provide a proof for this statement that lifts the Markovianity assumption as well—the central idea there is similar to what we showed here.

Embedding layer captures kinetic distances. In word embedding theory, the embedding layer provides a measure of similarity between words. However, from the path probability representation, it is unclear how the embedding layer works since the derivation can be done without embedding vectors $\mathbf{x}$. To have an understanding to Q_{lm} in the first-order Markov process, we first write the conditional probability $Q_{lm} = Q(\mathbf{x}_m^{(t+1)}|\mathbf{x}_l^{(t)})$ explicitly with softmax defined in Eq. (8) and embedding vectors $\mathbf{x}$ defined in Eq. (1):

$$\begin{aligned} Q_{lm} &= \frac{\exp(\mathbf{s}_m^{(t+1)} \cdot (\mathbf{D}_d \mathbf{h}^{(t)} + \mathbf{b}_d))}{\sum_k \exp(\mathbf{s}_k \cdot (\mathbf{D}_d \mathbf{h}^{(t)} + \mathbf{b}_d))} \\ &= \frac{\exp(\mathbf{s}_m^{(t+1)} \cdot (\mathbf{D}_d f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}) + \mathbf{b}_d))}{\sum_k \exp(\mathbf{s}_k \cdot (\mathbf{D}_d f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}) + \mathbf{b}_d))} \end{aligned} \tag{17}$$

where f is the recursive function $\mathbf{h}^{(t)} = f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)}) \approx f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)})$ which is defined with the update equation in Eq. (2)–(7). In Eq. (17), $\boldsymbol{\theta}$ denotes various parameters including all weight matrices and biases, and the summation index k runs over all possible states. Now we can use multivariable Taylor's theorem to approximate $f_{\boldsymbol{\theta}}$ as the linear term around a point $\mathbf{a}$ as long as $\mathbf{a}$ is not at any local minimum of $f_{\boldsymbol{\theta}}$:

$$f_{\boldsymbol{\theta}}(\mathbf{x}^{(t)}) \approx f_{\boldsymbol{\theta}}(\mathbf{a}) + \mathbf{A}_{\boldsymbol{\theta}}(\mathbf{x}^{(t)} - \mathbf{a}) \tag{18}$$

where $\mathbf{A}_{\boldsymbol{\theta}}$ is the L by M matrix defined to be $(\mathbf{A}_{\boldsymbol{\theta}})_{ij} = \frac{\partial (f_{\boldsymbol{\theta}})_i}{\partial x_j}|_{\mathbf{x}=\mathbf{a}}$. Then Eq. (17) becomes

$$Q_{lm} = \frac{\exp(C_m^{(t+1)}) \exp(\mathbf{s}_m^{(t+1)} \cdot \mathbf{D}_d \mathbf{A}_{\boldsymbol{\theta}} \mathbf{x}_l^{(t)})}{\sum_k \exp(C_k) \exp(\mathbf{s}_k \cdot \mathbf{D}_d \mathbf{A}_{\boldsymbol{\theta}} \mathbf{x}_l^{(t)})} \tag{19}$$

where $C_i^{(t+1)} = \mathbf{s}_i^{(t+1)} \cdot [\mathbf{D}_d(f_{\boldsymbol{\theta}}(\mathbf{a}_l) + \mathbf{A}_{\boldsymbol{\theta}}\mathbf{a}_l) + \mathbf{b}_d]$. We can see in Eq. (19) how the embedding vectors come into the transition probability. Specifically, there is a symmetric form between output one-hot vectors $\mathbf{s}_m^{(t+1)}$ and the input one-hot vectors $\mathbf{s}^{(t)}$, in which $\mathbf{x}^{(t)} = \mathbf{\Lambda}\mathbf{s}^{(t)}$ and $\mathbf{\Lambda}$ is the input embedding matrix, $\mathbf{D}_d\mathbf{A}_{\boldsymbol{\theta}}$ can be seen as the output embedding matrix, and $C_i^{(t+1)}$ is the correction of time lag effect. While we do not have an explicit way to calculate the output embedding matrix so defined, Eq. (19) motivates us to define the following ansatz for the transition probability:

$$Q_{lm} = Q(\mathbf{x}_m|\mathbf{x}_l) = \frac{\exp(\mathbf{x}_m \cdot \mathbf{x}_l)}{\sum_k \exp(\mathbf{x}_k \cdot \mathbf{x}_l)} \tag{20}$$

where $\mathbf{x}_m$ and $\mathbf{x}_l$ are both calculated by the input embedding matrix $\mathbf{\Lambda}$. The expression in Eq. (20) is thus a tractable approximation to the more exact transition probability in Eq. (19). Furthermore, we show through numerical examples of test systems that our ansatz for Q_{lm} does correspond to the kinetic connectivity between states. That is, the LSTM embedding layer with the transition probability through Eq. (20) can capture the average commute time between two states in the original physical system, irrespective of the quality of low-dimensional projection fed to the LSTM[25–27].

Test systems. To demonstrate our ideas, here we consider a range of different dynamical trajectories. These include three model potentials, the popular model molecule alanine dipeptide, and trajectory from single molecule force spectroscopy experiments on a multi-state riboswitch.[20] The sample trajectories of these test systems and the data preprocessing strategies are put in the Supplementary Note 5 and Supplementary Figs. 14–18 When applying our neural network to the model systems, the embedding dimension M is set to 8 and LSTM unit L set to 64. When learning trajectories for alanine dipeptide and riboswitch, we took $M = 128$ and $L = 1024$. All time series were batched into sequences with a sequence length of 100 and the batch size of 64. For each model potential, the neural network was trained using the method of stochastic gradient descent for 20 epochs until the training loss becomes smaller than the validation loss, which means an appropriate training has been reached. For alanine dipeptide, 40 training epochs were used. Our neural network was built using TensorFlow version 1.10. Further system details are provided in "Methods" section.

Boltzmann statistics and kinetics for model potentials. The first test we perform for our LSTM set-up is its ability to capture the Boltzmann weighted statistics for the different states in each model potential. This is the probability distribution P or equivalently the related free energy $F = -\frac{1}{\beta}\log \mathrm{P}$, and can be calculated by direct counting from the trajectory. As can be seen in Fig. 2, the LSTM does an excellent job of recovering the Boltzmann probability within error bars.

Next we describe our LSTM deals with a well-known problem in analyzing high-dimensional data sets through low-dimensional projections. One can project the high-dimensional data along many different possible low-dimensional order parameters, for instance x, y, or a combination thereof in Fig. 2. However most such projections will end up not being kinetically truthful and give a wrong impression of how distant the metastable states actually are from each other in the underlying high-dimensional space. It is in general hard to come up with a projection that preserves the kinetic properties of the high-dimensional space. Consequently, it is hard to design analysis or sampling methods that even when giving a time-series along a sub-optimal projection, still capture the true kinetic distance in the underlying high-dimensional space.

Here we show how our LSTM model is agnostic to the quality of the low-dimensional projection in capturing accurate kinetics. Given that for each of the 3 potentials the LSTM was provided only the x−trajectory, we can expect that the chosen model

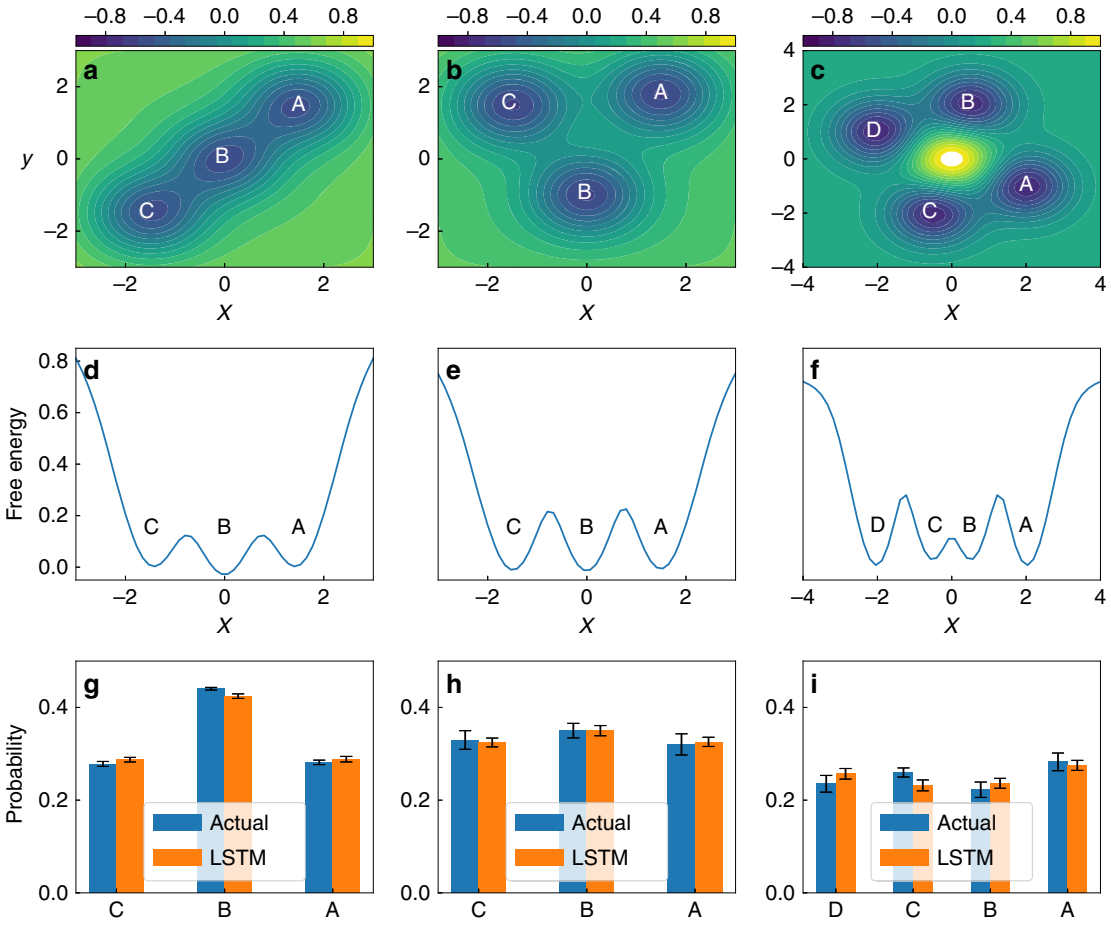

Fig. 2 Boltzmann statistics for model systems. The analytical free energy generated from **a** linear 3-state, **b** triangular 3-state, **c** symmetric 4-state model potentials and **d**, **e**, **f** are the corresponding 1-dimensional projections along *x*-direction. In the bottom, we compare the Boltzmann probabilities of **g** linear 3-state, **h** triangular 3-state, and **i** symmetric 4-state models for every labeled state generated from actual MD simulation and from our long short-term memory (LSTM) network. The errorbars are calculated as standard errors.

potentials constitute different levels of difficulties in generating correct kinetics. Specifically, a one-dimensional projection along *x* is kinetically truthful for the linear 3-state potential in Fig. 2a but not for the triangular 3-state and the 4-state potentials in Fig. 2b and c, respectively. For instance, Fig. 2e gives the impression that state C is kinetically very distant from state A, while in reality for this potential all 3 pairs of states are equally close to each other. Similar concerns apply to the 4-state potential.

In Figs. 3 and 4a–c and d–f we compare the actual versus LSTM-predicted kinetics for moving between different metastable states for different model potentials, for all pairs of transitions in both directions (i.e., for instance A to B and B to A). Specifically, Fig. 3a–c and 3d–f shows results for moving between the 3 pairs of states in the linear and triangular 3-state potentials, respectively. Figure 4 shows results for the 6 pairs of states in the 4-state potential. Furthermore, for every pair of state, we analyze the transition time between those states as a function of different minimum commitment or commit time, i.e., the minimum time that must be spent by the trajectory in a given state to be classified as having committed to it. A limiting value, and more specifically the rate at which the population decays to attain to such a limiting value, corresponds to the inverse of the rate constant for moving between those states[28,29]. Thus here we show how our LSTM captures not just the rate constant, but time-dependent fluctuations in the population in a given metastable state as equilibrium is attained. The results are averaged over 20 independent segments taken from the trajectories of different trials of training for the 3-state potentials and 10 independent segments for the 4-state potential.

As can be seen in Figs. 3 and 4, the LSTM model does an excellent job of reproducing well within errorbars the transition times between different metastable states for different model potentials irrespective of the quality of the low-dimensional projection. Firstly, our model does tell the differences between linear and triangular 3-state models (Fig. 3) even though the projected free energies along the *x* variable input into LSTM are same (Fig. 2). The number of transitions between states A and C is less than the others; while for triangular configuration, the numbers of transitions between all pairs of states are similar. The rates at which the transition count decays as a function of commitment time is also preserved between the input data and the LSTM prediction.

The next part of our second test is the 4-state model potential. In Fig. 4 we show comparisons for all 6 pairs of transitions in both forward and reverse directions. A few features are immediately striking here. Firstly, even though states B and C are perceived to be kinetically proximal from the free energy (Fig. 2), the LSTM captures that they are distal from each other and correctly assigns similar kinetic distance to the pairs B, C as it does to A, D. Secondly, there is asymmetry between the forward and backward directions (for e.g., A to D and D to A, indicating that the input trajectory itself has not yet sufficiently sampled the

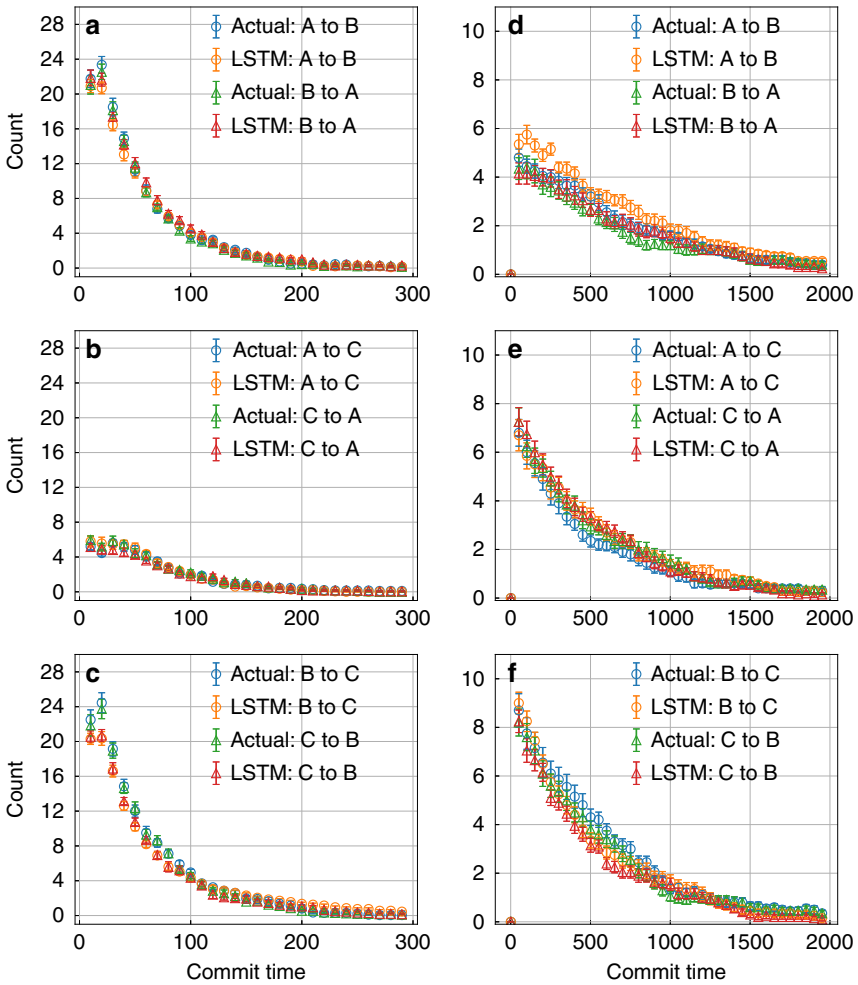

Fig. 3 Kinetics for 3-state model systems. Number of transitions between different pairs of metastable states as a function of commitment time defined in "Results" section. The calculations for linear and triangular configurations are shown in **a**–**c** and **d**–**f**, respectively. Error bars are illustrated and were calculated as standard errors.

slow transitions in this potential. As can be seen from Fig. 2c the input trajectory has barely 1 or 2 direct transitions for the very high barrier A to D or B to C. This is a likely explanation for why our LSTM model does a bit worse than in the other two model potentials in capturing the slowest transition rates, as well as the higher error bars we see here. In other words, so far we can conclude that while our LSTM model can capture equilibrium probabilities and transition rates for different model potentials irrespective of the input projection direction or order parameter, it is still not a panacea for insufficient sampling itself, as one would expect.

Boltzmann statistics and kinetics for alanine dipeptide. Finally, we apply our LSTM model to the study of conformational transitions in alanine dipeptide, a model biomolecular system comprising 22 atoms, experiencing thermal fluctuations when coupled to a heat bath. The structure of alanine dipeptide is shown in Fig. 5a. While the full system comprises around 63 degrees of freedom, typically the torsional angles ϕ and ψ are used to identify the conformations of this peptide. Over the years a large number of methods have been tested on this system in order to perform enhanced sampling of these torsions, as well as to construct optimal reaction coordinates[30–33]. Here we show that our LSTM model can very accurately capture the correct Boltzmann statistics, as well as transition rates for moving between the two dominant metastable states known as C_{7eq} and C_{7ax}. Importantly, the reconstruction of the equilibrium probability and transition kinetics, as shown in Fig. 5 and Table 1 is extremely accurate irrespective of the choice of one-dimensional projection time series fed into the LSTM. Specifically, we do this along $\sin\phi$ and $\sin\psi$, both of which are known to quite distant from an optimized kinetically truthful reaction coordinate[19,34], where again we have excellent agreement between input and LSTM-predicted results.

Learning from single molecule force spectroscopy trajectory. In this section, we use our LSTM model to learn from single molecule force spectroscopy experiments of a multi-state riboswitch performed with a constant force of 10.9 pN. The data points are measured at 10 kHz (i.e., every 100 μs). Other details of the experiments can be found in ref. [20]. The trajectory for a wide range of extensions starting 685 nm up to 735 nm was first spatially discretized into 34 labels, and then converted to a time series of one hot vectors, before being fed into the LSTM model. The results are shown in Fig. 6. In Fig. 6a, we have shown an agreement between a profile of probability density averaged over 5 independent training sets with the probability density calculated from the experimental data. Starting from the highest extension, the states are fully unfolded (U), longer intermediate (P3) and shorter intermediate (P2P3)[20]. From Fig. 6b–c, we see that the LSTM model captures the kinetics for moving between all 3 pairs of states for a very wide range of commitment times.

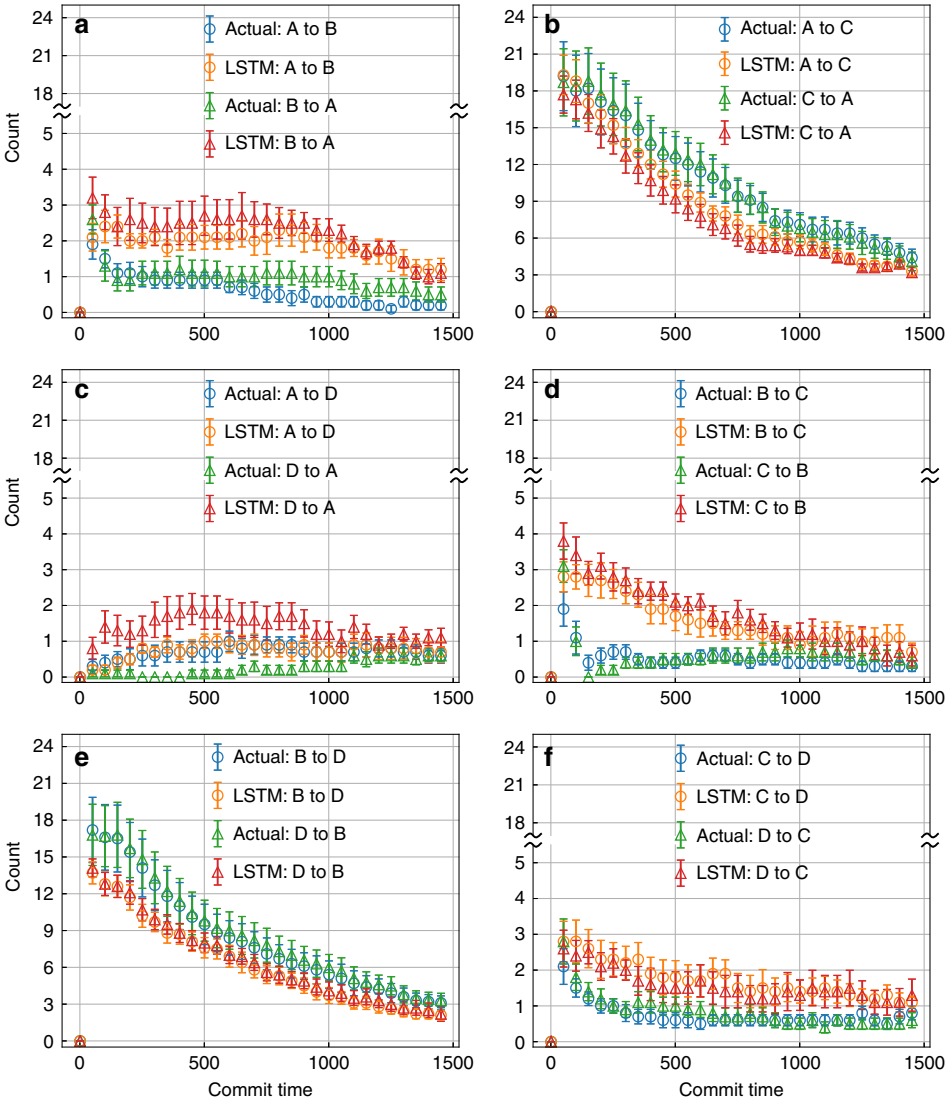

Fig. 4 Kinetics for 4-state model system. Number of transitions between different pairs of metastable states as a function of commitment time defined in "Results" section for 4-state model system. Error bars are illustrated and were calculated as standard errors.

Embedding layer based kinetic distance. In Eq. (19), we derived a non-tractable relation for conditional transition probability in the embedding layer, and then through Eq. (20) we introduced a tractable ansatz in the spirit of Eq. (19). Here we revisit and numerically validate Eq. (20). Specifically, given any two embedding vectors $\mathbf{x}_l$ and $\mathbf{x}_m$ calculated from any two states l and m, we estimate the conditional probability Q_{lm} using Eq. (20). We use Q_i to denotes the Boltzmann probability predicted by the LSTM model. We then write down the interconversion probability k_{lm} between states l and m as:

$$k_{lm} = Q_l Q_{lm} + Q_m Q_{ml} \equiv 1/t_{lm} \tag{21}$$

From inverting this rate we then calculate an LSTM-kinetic time as $t_{lm} \equiv 1/k_{lm} = 1/(Q_l Q_{lm} + Q_m Q_{ml})$. In Fig. 7, we compare t_{lm} with the actual transition time τ_{lm} obtained from the input data, defined as

$$\tau_{lm} = T/\langle N_{lm} \rangle \tag{22}$$

Here N_{lm} is the mean number of transitions between state l and m. As this number varies with the precise value of commitment time, we average N_{lm} over all commit times to get $\langle N_{lm} \rangle$. These two timescales t_{lm} and τ_{lm} thus represent the average commute time or kinetic distance[25,26] between two states l and m. To facilitate the comparison between these two very differently derived timescales or kinetic distances, we rescale and shift them to lie between 0 and 1. The results in Fig. 7 show that the embedding vectors display the connectivity corresponding to the original high-dimensional configuration space rather than those corresponding to the one-dimensional projection. The model captures the correct connectivity by learning kinetics, which is clear evidence that it is able to bypass the projection error along any degree of freedom. The result also explains how is it that no matter what degree of freedom we use, our LSTM model still gives correct transition times. As long as the degree of freedom we choose to train the model can be used to discern all metastable states, we can even use Eq. (20) to see the underlying connectivity. Therefore, the embedding vectors in LSTM can define a useful distance metric which can be used to understand and model dynamics, and are possibly part of the reason why LSTMs can model kinetics accurately inspite of quality of projection and associated non-Markvoian effects.

Comparing with Markov state model and Hidden Markov Model. In this section, we briefly compare our LSTM model with

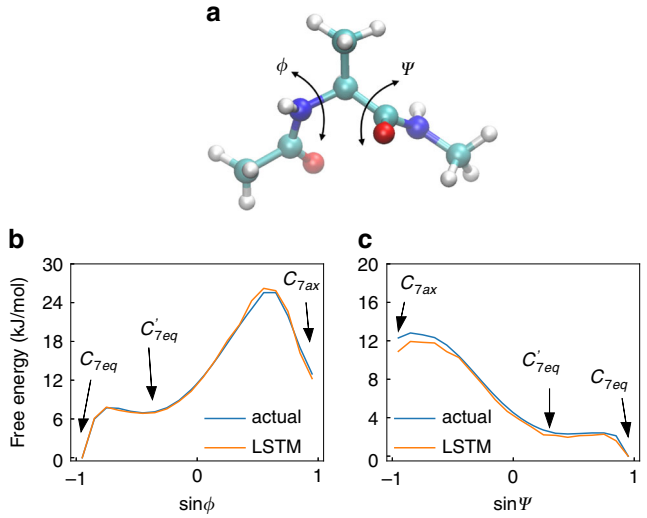

Fig. 5 Boltzmann statistics for alanine dipeptide. a The molecular structure of alanine dipeptide used in the actual MD simulation. The torsional angles ϕ and ψ as the collective variables (CVs) are shown. **b** and **c** The 1-dimensional free energy curves along sin ϕ and sin ψ are calculated using actual MD data and the data generated from LSTM. For the calculation of a different epoch, please see Supplementary Note 2 and Supplementary Table 1.

Table 1 Kinetics for alanine dipeptide.

Alanine dipeptide

CVs	Label	C_{7eq} to C_{7ax} (ps)	C_{7ax} to C_{7eq} (ps)
sin ϕ	Actual	5689.22 ± 962.366	107.93 ± 11.267
	LSTM	5752.16 ± 710.399	103.81 ± 14.268
sin ψ	Actual	5001.42 ± 643.943	105.70 ± 13.521
	LSTM	4325.01 ± 526.293	81.68 ± 10.288

Inverse of transition rates for conformational transitions in alanine dipetide calculated from actual MD trajectories of LSTM model. Here we show the calculation along two different CVs: sin ϕ and sin ψ.

standard approaches for building kinetic models from trajectories, namely the Markov state model (MSM)[35] and Hidden Markov model (HMM)[36–38]. Compared to LSTM, the MSM and HMM have smaller number of parameters, making them faster and more stable for simpler systems. However, both MSM and HMM require choosing an appropriate number of states and lag time[35,38,39]. Large number of pre-selected states or small lag time can lead to non-Markovian behavior and result in an incorrect prediction. Even more critically, choosing a large lag time also sacrifices the temporal precision. On the other hand, there is no need to determine the lag time and number of states using the LSTM network because LSTM does not rely on the Markov property. Choosing hyperparameters such as M and L may be comparable to choosing number of hidden states for HMM, while very similar values of M and L worked for systems as different as MD trajectory of alanine dipeptide and single molecule force spectroscopy trajectory of a riboswitch. At the same time, LSTM always generates the data points with the same temporal precision as it has in the training data irrespective of the intrinsic timescales it learns from the system. In Fig. 8, we provide the results of using HMM and MSM for the riboswitch trajectory with the same binning method and one-hot encoded input, to be contrasted with similar plots using LSTM in Fig. 6. Indeed both MSM and HMM achieve decent agreement with the true kinetics only if the commit time is increased approximately beyond 10 ms, while LSTM as shown in Fig. 6 achieved perfect agreement for all commit times. From this figure, it can be seen that the LSTM model achieves an expected agreement with as fine of a temporal precision as desired, even though we use 20 labels for alanine dipeptide and 34 labels for experimental data to represent the states. The computational efforts needed for the various approaches (LSTM, MSM, and HMM) are also provided in the Supplementary Note 3 and Supplementary Table 2–3, where it can be seen that LSTM takes similar amount of effort as HMM. The package we used to build the MSM and HMM is PyEMMA with version 2.5.6[40]. The models were built with lag time = 0.5 ms for MSM and lag time = 3 ms for HMM, where the HMM were built with number of hidden states = 3. A more careful comparison of the results along with analyses with other parameter choices such as different number of hidden states for HMM are provided in the Supplementary Note 4 and Supplementary Figs. 1–13, where we find all of these trends to persist.

Discussion

In summary we believe this work demonstrates potential for using AI approaches developed for natural language processing such as speech recognition and machine translation, in unrelated domains such as chemical and biological physics. This work represents a first step in this direction, wherein we used AI, specifically LSTM flavor of recurrent neural networks, to perform kinetic reconstruction tasks that other methods[41,42] could have also performed. We would like to argue that demonstrating the ability of AI approaches to perform tasks that one could have done otherwise is a crucial first step. In future works we will exploring different directions in which the AI protocol developed here could be used to perform tasks which were increasingly non-trivial in non-AI setups. More specifically, in this work we have shown that a simple character-level language model based on LSTM neural network can learn a probabilistic model of a time series generated from a physical system such as an evolution of Langevin dynamics or MD simulation of complex molecular models. We show that the probabilistic model can not only learn the Boltzmann statistics but also capture a large spectrum of kinetics. The embedding layer which is designed for encoding the contextual meaning of words and characters displays a nontrivial connectivity and has been shown to correlate with the kinetic map defined for reversible Markov chains[25,26,43]. An interesting future line of work for the embedding layer can be to uncover different states when they are incorrectly represented by the same reaction coordinate value, which is similar to finding different contextual meaning of the same word or character. For different model systems considered here, we could obtain correct timescales and rate constants irrespective of the quality of order parameter fed into the LSTM. As a result, we believe this kind of model outperforms traditional approaches for learning thermodynamics and kinetics, which can often be very sensitive to the choice of projection. Finally, the embedding layer can be used to define a new type of distance metric for high-dimensional data when one has access to only some low-dimensional projection. We hope that this work represents a first step in the use of RNNs for modeling, understanding and predicting the dynamics of complex stochastic systems found in biology, chemistry and physics.

Methods

Model potential details. All model potentials have two degrees of freedom x and y. Our first two models (shown in Fig. 2a and b) have three metastable states with governing potential $U(x, y)$ given by

$$U(x,y) = W(x^6 + y^6) - G(x,x_1)G(y,y_1) - G(x,x_2)G(y,y_2) - G(x,x_3)G(y,y_3) \quad (23)$$

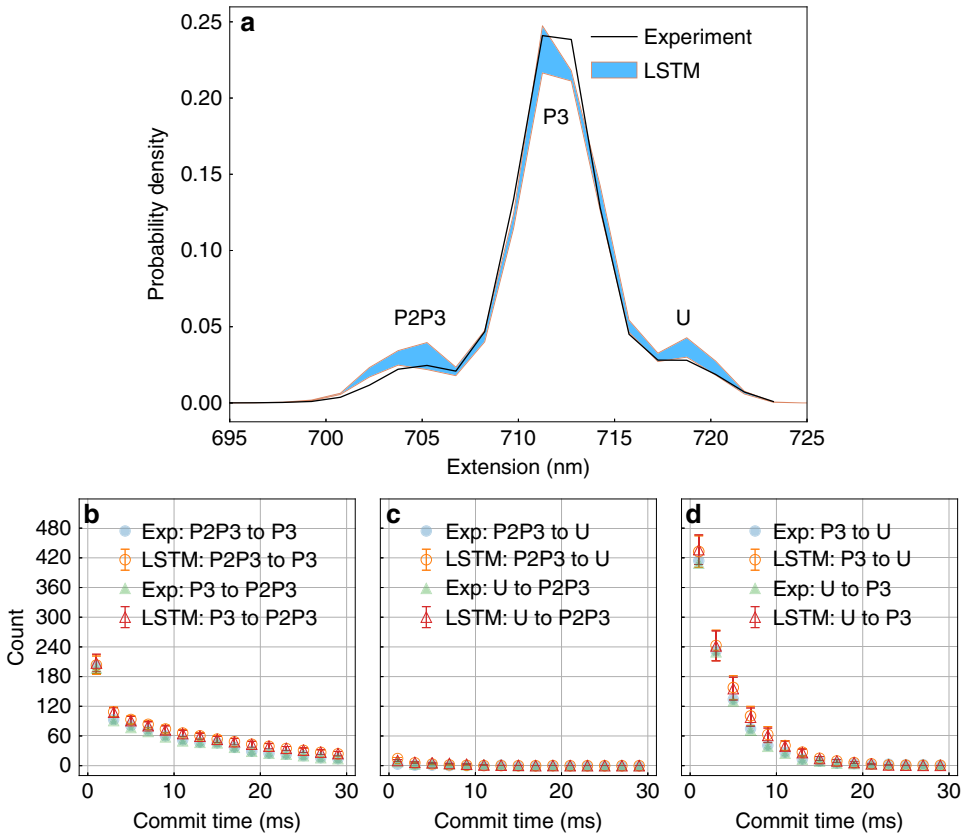

Fig. 6 Boltzmann statistics and kinetics for riboswitch. Using LSTM model to learn thermodynamics and kinetics from a folding and unfolding trajectory taken from a single molecule force spectroscopy measurement[20]: **a** Comparison between the probability density learned by the LSTM model and calculated from the experimental data. The regions between errorbars defined as standard errors are filled with blue color. **b**–**d** Commit time plots calculated by counting the transitions in the trajectory generated by LSTM and the experimental trajectory. The commit time is the minimum time that must be spent by the trajectory in a given state to be classified as having committed to it. Error bars are illustrated and were calculated as standard errors.

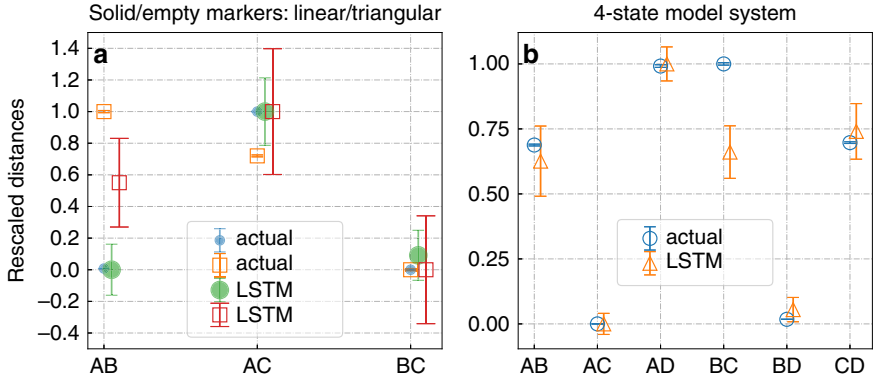

Fig. 7 Analysis of embedding layers for model systems. Our analysis of the embedding layer constructed for **a** the linear and triangular 3-state and **b** the 4-state model systems. In **a**, we use solid circle and empty square markers, respectively to represent linear and triangular 3-state model potentials. In each plot, the data points are shifted slightly to the right for clarity. The distances marked actual and LSTM represent rescaled mean transition times as per Eqs. (22) and (21), respectively. Error bars were calculated as standard errors over 50 different trajectories.

where $W = 0.0001$ and $G(x, x_0) = e^{-\frac{(x-x_0)^2}{2\sigma^2}}$ denotes a Gaussian function centered at x_0 with width $\sigma = 0.8$. We also build a 4-state model system with governing interaction potential:

$$\begin{aligned} U(x,y) = {} & W(x^4 + y^4) + G(x,0.0)G(y,0.0) \\ & - G(x,2.0)G(y,-1.0) - G(x,0.5)G(y,2.0) \\ & - G(x,-0.5)G(y,-2.0) - G(x,-2.0)G(y,1.0) \end{aligned} \quad (24)$$

The different local minima corresponding to the model potentials in Eq. (23) and Eq. (24) are illustrated in Fig. 2. We call these as linear 3-state, triangular 3-state, and 4-state models, respectively. The free energy surfaces generated from the simulation of Langevin dynamics[44] with these model potentials are shown in Fig. 2a–c.

Molecular dynamics details. The integration timestep for the Langevin dynamics simulation was 0.01 units, and the simulation was performed at $\beta =$ 9.5 for linear 3-state and 4-state potentials and $\beta = 9.0$ for triangular 3-state potential, where $\beta = 1/k_BT$. The MD trajectory for alanine dipeptide was obtained using the software GROMACS 5.0.4[45,46], patched with PLUMED 2.4[47]. The temperature was kept constant at 450 K using the velocity rescaling thermostat[48].

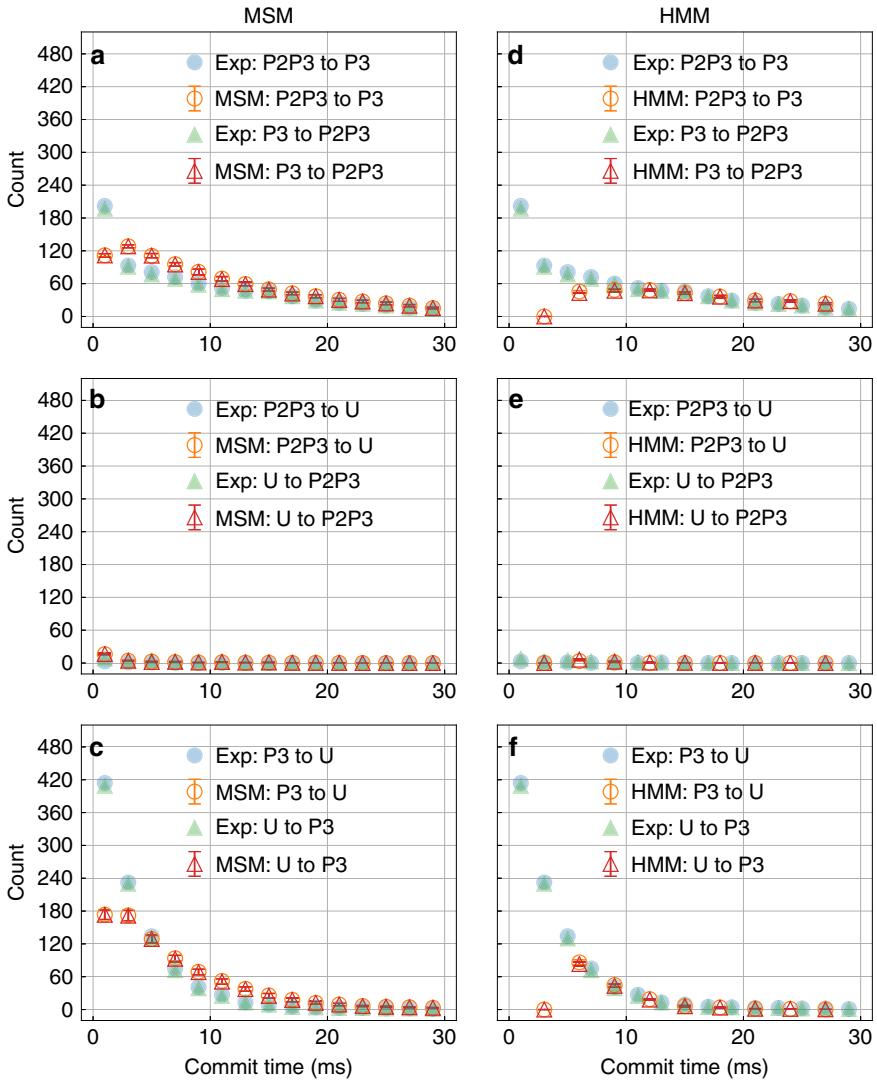

Fig. 8 Riboswitch kinetics through alternate approaches. Number of transitions between different pairs of metastable states as a function of commitment time defined in "Results" section for the single molecule spectroscopy trajectory as learned by MSM (left column) and HMM (right column). Associated error bars calculated as standard errors are also provided.

Data availability

The single-molecule force spectroscopy experiment data for riboswitch was obtained from the authors of ref. [20] and they can be contacted for the same. All the other data associated with this work is available from the corresponding author on request.

Code availability

MSM and HMM analyses were conducted with PyEMMA version 2.5.6.[40] and available at http://www.pyemma.org. A Python based code of the LSTM language model is implemented using keras (https://keras.io/) with tensorflow-gpu (https://www.tensorflow.org/) as a backend, and available for public use at https://github.com/tiwarylab/LSTM-predict-MD.

Received: 4 May 2020; Accepted: 23 September 2020;
Published online: 09 October 2020

References

1. Rico-Martinez, R., Krischer, K., Kevrekidis, I., Kube, M. & Hudson, J. Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data. *Chem. Engg. Commun.* **118**, 25–48 (1992).
2. Gicquel, N., Anderson, J. & Kevrekidis, I. Noninvertibility and resonance in discrete-time neural networks for time-series processing. *Phys. Lett. A* **238**, 8–18 (1998).
3. Graves, A., Liwicki, M., Fernández, S., Bertolami, R. & Bunke, H. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern. Anal. Mach. Intell.* **31**, 855–868 (2008).
4. Graves, A., Mohamed, A.-r. & Hinton, G. Speech recognition with deep recurrent neural networks. In *International Conference on Acoustics, Speech, and Signal Processing*. 6645–6649 (2013).
5. Cho, K., Van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bahdanau, D. & Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734 (2014).
6. Xingjian, S., Chen, Z., Wang, H. & Woo, W.-c. Convolutional lstm network: a machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*. 802–810 (2015).
7. Chen, K., Zhou, Y. & Dai, F. A LSTM-based method for stock returns prediction: a case study of china stock market. In *IEEE International Conference on Big Data*. 2823–2824 (2015).
8. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neur. Comp.* **9**, 1735–1780 (1997).
9. Sundermeyer, M., Schlüter, R. & Ney, H. LSTM neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*. (2012).
10. Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O. & Zaremba, W. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 11–19 (2014).

11. Hochreiter, S. et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (2001).
12. Agar, J. C., Naul, B., Pandya, S. & van Der Walt, S. Revealing ferroelectric switching character using deep recurrent neural networks. *Nat. Commun.* **10**, 1–11 (2019).
13. Eslamibidgoli, M. J., Mokhtari, M. & Eikerling, M. H. Recurrent neural network-based model for accelerated trajectory analysis in aimd simulations. Preprint at https://arxiv.org/abs/1909.10124 (2019).
14. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comp. Sci. Rev.* **3**, 127–149 (2009).
15. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**, 024102 (2018).
16. Noé, F., Olsson, S., Köhler, J. & Wu, H. Boltzmann generators: sampling equilibrium states of many-body systems with deep learning. *Science* **365**, eaaw1147 (2019).
17. Sidky, H., Chen, W. & Ferguson, A. L. Molecular latent space simulators. *Chem. Sci.* **11**, 9459–9467 (2020).
18. Bussi, G. & Laio, A. Using metadynamics to explore complex free-energy landscapes. *Nat. Rev. Phys.* **2**, 200–212 (2020).
19. Wang, Y., Ribeiro, J. M. L. & Tiwary, P. Past–future information bottleneck for sampling molecular reaction coordinate simultaneously with thermodynamics and kinetics. *Nat. Commun.* **10**, 1–8 (2019).
20. Neupane, K., Yu, H., Foster, D. A., Wang, F. & Woodside, M. T. Single-molecule force spectroscopy of the add adenine riboswitch relates folding to regulatory mechanism. *Nucl. Acid. Res.* **39**, 7677–7687 (2011).
21. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT press, 2016).
22. Cover, T. M. & Thomas, J. A. *Elements of Information Theory* (John Wiley & Sons, 2012).
23. Pressé, S., Ghosh, K., Lee, J. & Dill, K. A. Principles of maximum entropy and maximum caliber in statistical physics. *Rev. Mod. Phys.* **85**, 1115 (2013).
24. Moore, C. C. Ergodic theorem, ergodic theory, and statistical mechanics. *Proc. Natl Acad. Sci. USA* **112**, 1907–1911 (2015).
25. Noe, F., Banisch, R. & Clementi, C. Commute maps: separating slowly mixing molecular configurations for kinetic modeling. *J. Chem. Theor. Comp.* **12**, 5620–5630 (2016).
26. Noé, F. & Clementi, C. Kinetic distance and kinetic maps from molecular dynamics simulation. *J. Chem. Theor. Comp.* **11**, 5002–5011 (2015).
27. Tsai, S.-T. & Tiwary, P. On the distance between A and B in molecular configuration space. *Mol. Sim.* **46**, 1–8 (2020).
28. Hänggi, P., Talkner, P. & Borkovec, M. Reaction-rate theory: fifty years after kramers. *Rev. Mod. Phys.* **62**, 251 (1990).
29. Berne, B. J., Borkovec, M. & Straub, J. E. Classical and modern methods in reaction rate theory. *J. Phys. Chem.* **92**, 3711–3725 (1988).
30. Valsson, O., Tiwary, P. & Parrinello, M. Enhancing important fluctuations: rare events and metadynamics from a conceptual viewpoint. *Ann. Rev. Phys. Chem.* **67**, 159–184 (2016).
31. Salvalaglio, M., Tiwary, P. & Parrinello, M. Assessing the reliability of the dynamics reconstructed from metadynamics. *J. Chem. Theor. Comp.* **10**, 1420–1425 (2014).
32. Ma, A. & Dinner, A. R. Automatic method for identifying reaction coordinates in complex systems. *J. Phys. Chem. B* **109**, 6769–6779 (2005).
33. Bolhuis, P. G., Dellago, C. & Chandler, D. Reaction coordinates of biomolecular isomerization. *Proc. Natl Acad. Sci. USA* **97**, 5877–5882 (2000).
34. Smith, Z., Pramanik, D., Tsai, S.-T. & Tiwary, P. Multi-dimensional spectral gap optimization of order parameters (sgoop) through conditional probability factorization. *J. Chem. Phys.* **149**, 234105 (2018).
35. Husic, B. E. & Pande, V. S. Markov state models: from an art to a science. *J. Am. Chem. Soc.* **140**, 2386–2396 (2018).
36. Eddy, S. R. What is a hidden markov model? *Nat. Biotechnol.* **22**, 1315–1316 (2004).
37. McKinney, S. A., Joo, C. & Ha, T. Analysis of single-molecule fret trajectories using hidden markov modeling. *Bioph. Jour.* **91**, 1941–1951 (2006).
38. Blanco, M. & Walter, N. G. Analysis of complex single-molecule fret time trajectories. In *Methods in Enzymology*, Vol. 472, 153–178 (Elsevier, 2010).
39. Bowman, G. R., Beauchamp, K. A., Boxer, G. & Pande, V. S. Progress and challenges in the automated construction of markov state models for full protein systems. *J. Chem. Phys.* **131**, 124101 (2009).
40. Scherer, M. K. et al. Pyemma 2: a software package for estimation, validation, and analysis of markov models. *J. Chem. Theor. Comp.* **11**, 5525–5542 (2015).
41. Pérez-Hernández, G., Paul, F., Giorgino, T., De Fabritiis, G. & Noé, F. Identification of slow molecular order parameters for markov model construction. *J. Chem. Phys.* **139**, 07B604_1 (2013).
42. Chodera, J. D. & Noé, F. Markov state models of biomolecular conformational dynamics. *Curr. Op. Struc. Bio. y.* **25**, 135–144 (2014).
43. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119 (2013).
44. Bussi, G. & Parrinello, M. Accurate sampling using langevin dynamics. *Phys. Rev. E* **75**, 056707 (2007).
45. Berendsen, H. J., van der Spoel, D. & van Drunen, R. Gromacs: a message-passing parallel molecular dynamics implementation. *Comp. Phys. Commun.* **91**, 43–56 (1995).
46. Abraham, M. J. et al. Gromacs: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **1**, 19–25 (2015).
47. Bonomi, M., Bussi, G. & Camilloni, C. C. Promoting transparency and reproducibility in enhanced molecular simulations. *Nat. Methods* **16**, 670–673 (2019).
48. Bussi, G., Donadio, D. & Parrinello, M. Canonical sampling through velocity rescaling. *J. Chem. Phys.* **126**, 014101 (2007).

Acknowledgements

P.T. thanks Dr. Steve Demers for suggesting the use of LSTMs. The authors thank Carlos Cuellar for the help in early stages of this project, Michael Woodside for sharing the single molecule trajectory with us, Yihang Wang for in-depth discussions, Dedi Wang, Yixu Wang, Zachary Smith for their helpful insights and suggestions. Acknowledgment is made to the Donors of the American Chemical Society Petroleum Research Fund for partial support of this research (PRF 60512-DNI6). We also thank Deepthought2, MARCC and XSEDE (projects CHE180007P and CHE180027P) for computational resources used in this work.

Author contributions

P.T., S.T., and E.K. designed research; P.T., S.T., and E.K. performed research; S.T. analyzed data; S.T. and P.T. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41467-020-18959-8.

Correspondence and requests for materials should be addressed to P.T.

Peer review information *Nature Communications* thanks Simon Olsson and the other anonymous reviewer(s) for their contribution to the peer review of this work.

Reprints and permission information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.