A Deep-Learning Approach to the Dynamics of Landau–Zenner Transitions

A deep-learning approch to the dynamics of Landau-Zenner transitions

Linliang Gao[1,2], Kewei Sun[1*], Huiru Zheng[3], and Yang Zhao [2 †]
[1] *School of Science, Hangzhou Dianzi University, Hangzhou 310018, China*
[2] *Division of Materials Science, Nanyang Technological University, Singapore 639798, Singapore*
[3] *School of Computing, Ulster University at Jordanstown, Newtownabbey, Co. Antrim, United Kingdom BT37 0QB*
(Dated: April 21, 2021)

Traditional approaches to the dynamics of the open quantum systems with high precision are often resources consuming. How to improve computation accuracy and efficiency for target systems presents us with one of the most difficult challenges. In this work, combining unsupervised and supervised learning algorithms, a deep-learning approach is introduced to simulate and predict Landau-Zenner dynamics. Data obtained from the multiple Davydov D_2 Ansatz with a low multiplicity of four are used for training, while the data from the trial state with a high multiplicity of ten are adopted as target data to assess the accuracy of prediction. After proper training, our method can successfully predict and simulate Landau-Zenner dynamics using only random noise and two adjustable model parameters. Compared to the high-precision dynamics data from the multiple Davydov D_2 Ansatz with a multiplicity of ten, the error rate falls below 0.6 %.

I. INTRODUCTION

As one of the most fundamental phenomena in quantum dynamics, the Landau-Zener (LZ) transition happens as a result of propagating through the level crossing between diabatic surfaces at a constant sweeping speed. Its final transition probability obtained first by Landau and Zener in 1932 [1, 2], and the LZ transition is found to play an important role in a variety of fields, including atomic and molecular physics [3–5], quantum optics [7], chemical physics [6], and quantum information science [8]. Recently, there has obtained renewed interest in the LZ transition with the advent of a huge variety of new applications [9, 10], such as a nitrogen-vacancy center spin in isotopically purified diamond [11], a microwave driven superconducting qubit coupled to a two-level system, [12] and a spin-orbit-coupled Bose-Einstein condensate [13]. Recent advances in circuit quantum electrodynamics (QED) devices make them promising candidates for exploration of the LZ transitions due to their potential scalability and wide-ranging tunable parameters [14–19]. Circuit QED is an implementation of cavity QED in superconducting quantum circuits. A superconducting flux qubit coupled to a quantum interference device [16] has been fabricated by Chiorescu *et al.*, as well as a charge qubit coupled to a transmission line resonator by Wallraff *et al.* [17]. Similar developments have paved the way to implement the LZ transition experiments where the energy gap between the diabatic states can be tuned by external fields [18].

A schematic diagram of the LZ transition is shown in Fig. 1, where $|G\rangle$ and $|E\rangle$ represent the adiabatic states, and $|\uparrow\rangle$ and $|\downarrow\rangle$, the diabatic states. In general, a qubit cannot be completely isolated from its surrounding environment. To accurately describe the dynamics of the qubit, the environmental influence on the qubit must be taken into account[20–24]. In this context, the multiple Davydov trial states have been previously utilized to accurately simulate the dissipative dynamics of LZ model coupled to a bosonic bath in the framework of the standard time-dependent variation [25].

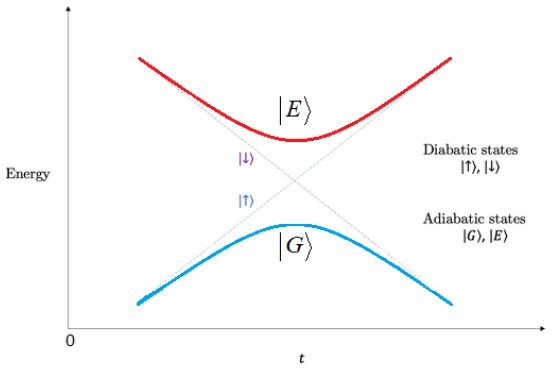

Figure 1: Energy-level evolution in the adiabatic and diabatic states of a LZ transition.

Development of artificial intelligence nowadays is increasingly more influential to our daily lives, and numerous applications are found in a variety of fields such as machine vision, fingerprint/face/retina/iris recognition [26, 27], expert systems [28], automatic planning, genetic programming, intelligent search, intelligent control, robotics, linguistics [29, 30] and image understanding [31]. In this work, three machine learning methods, namely, an unsupervised learning algorithm using the generative adversarial network (GAN), and two algorithms of supervised learning, namely, the convolutional neural network (CNN) and the back propagation neural network (BPNN), are integrated to predict the occurrence probability of LZ transitions, after the initial

*Electronic address: skw79724@163.com
†Electronic address: YZhao@ntu.edu.sg

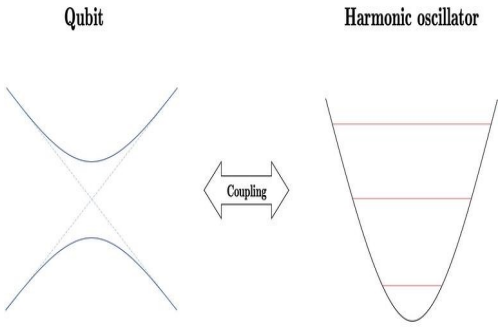

Figure 2: The qubit is coupled to a single harmonic oscillator.

set of training data is established by the multiple Davydov D_2 ansatz via time-dependent variation. GAN as a class of machine learning frameworks in which two neural networks (NN) compete with each other in a zero-sum game, has been extremely successful in artistic creation and image synthesis. For example, GAN can be used to migrate makeup style from a given reference face image to another non-makeup face image without the change of face identity [32]. CNN is a class of deep NNs often applied to imagery analysis, while BPNN, a basic neural network, allows feedbacks to the network to self-correct the network parameters.

The rest of the paper is organized as follows. In Sec. II, we introduced the Hamiltonian and the machine learning algorithms that could be applied in the study of LZ dynamics. In Sec. III, results from the GAN, CNN and BPNN are presented, and the prediction accuracy of the present method within a certain range is analyzed. Conclusions are drawn in Sec. IV.

II. METHODOLOGY

A. The Hamiltonian

For a time-dependent quantum system coupled with a single bosonic mode, its Hamiltonian can often be partitioned into three parts,

$$H = H_{\rm S} + H_{\rm B} + H_{\rm SB}. \tag{1}$$

Here, the system Hamiltonian is given by the standard LZ Hamiltonian, that is

$$H_{\rm S} = H_{\rm LZ} = \nu t\frac{\sigma_z}{2} + \Delta\frac{\sigma_x}{2}, \tag{2}$$

where σ_z and σ_x are the Pauli matrices, and $\nu > 0$ is sweeping speed of level crossing. The tunneling strength Δ represents the internal interaction between diabatic states. In our calculations, $\nu = 0.01$ is adopted.

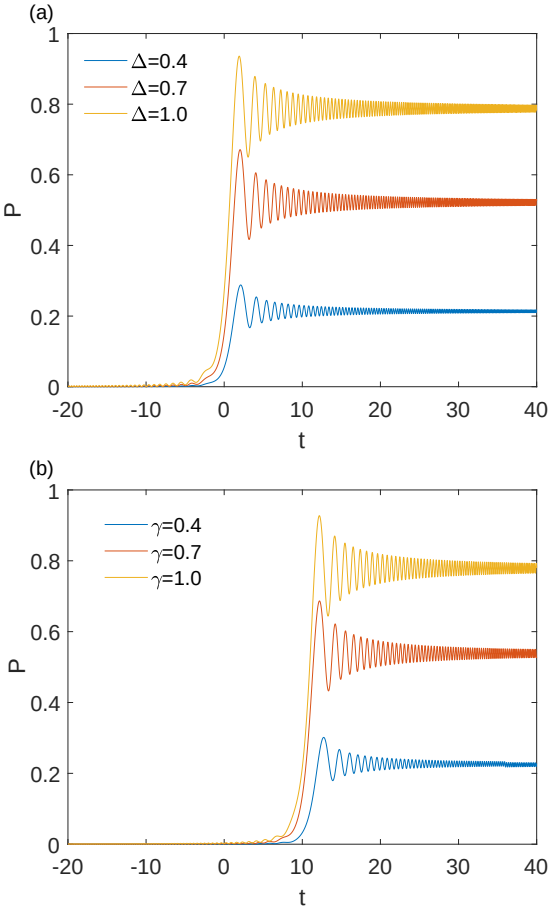

Figure 3: **There are two mechanisms to form a two-stage picture for the time evolution of the LZ transition probabilities. (a) The first transition stage is induced by direct tunneling D between the two levels with off-diagonal coupling strength $\gamma = 0$. From top down, tunneling strength $\Delta = 1.0\sqrt{\nu/\hbar}$, $0.7\sqrt{\nu/\hbar}$, $0.4\sqrt{\nu/\hbar}$, named the standard LZ transition. (b) The second transition stage results from the indirect off-diagonal coupling to the single-oscillator mode with tunneling strength $\Delta = 0$. From top down, off-diagonal coupling strength $\gamma = 1.0\sqrt{\nu/\hbar}, 0.7\sqrt{\nu/\hbar}, 0.4\sqrt{\nu/\hbar}$.**

As shown in Fig. 2, only a single harmonic oscillator is included in the bath Hamiltonian

$$H_{\rm B} = \hbar\omega b^\dagger b, \tag{3}$$

where $\hbar = 1$ is assumed throughout, ω represents the frequency of the phonon mode, and $b^\dagger$ (b) is the generation (annihilation) operator. The system-bath interaction Hamiltonian $H_{\rm SB}$ is assumed to be off-diagonal

$$H_{\rm SB} = \frac{\gamma}{2}\sigma_x(b + b^\dagger), \tag{4}$$

where γ is the non-diagonal coupling strength.

An alternative to the multiple Davydov D_1 trial state [33], the multiple D_2 Ansatz (also known as the multi-D_2

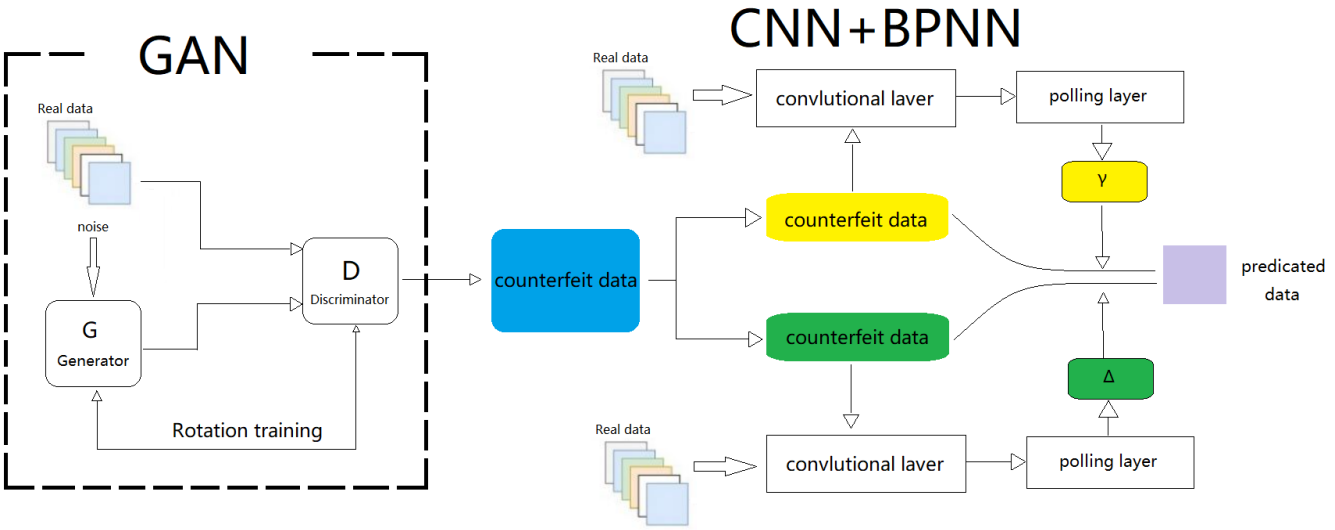

Figure 4: Overall framework of the proposed ML approach, which can be regarded as a combination of one GAN, two CNNs, and multiple BPNNs. GAN uses random noise and "Real data" to generate a set of counterfeit data. CNNs are adopted to identify the parameter info imbedded in the counterfeit data. Finally, the BPNNs are used to establish a mapping relationship for each part to error reduction.

Ansatz) with multiplicity M, can be written as

$$|\psi^M_{\mathrm{D}_2}(t)\rangle = \sum_m |m\rangle \sum_{k=1}^{M} A_{m,k}(t) e^{\left(\sum_q f_{k,q}(t) b_q^\dagger - H.c.\right)} |0\rangle,$$

where $|m\rangle$ denotes the spin state $|\uparrow\rangle$ or $|\downarrow\rangle$, $H.c.$ indicates the Hermitian conjugate, and $|0\rangle$ is the vacuum state of the bosonic bath. The time-dependent variational parameters $\{A_{m,k}, f_{k,q}\}$ are determined by the Dirac-Frenkel time-dependent variational principle. The reader is referred to Ref. [25] for calculation details.

The time-dependent transition probability is given by

$$P_{\uparrow\to\downarrow}(t) = \langle\psi^M_{\mathrm{D}_2}(t)|\uparrow\rangle\langle\downarrow|\psi^M_{\mathrm{D}_2}(t)\rangle. \tag{5}$$

As an illustration, the LZ transition probabilities $P_{\uparrow\to\downarrow}(t)$ for different control parameters, obtained with the multiple D_2 anstaz with $M = 10$, are shown in Fig. 3, where we set $\omega = 10\sqrt{\nu/\hbar}$.

In the long-time limit, the probability of LZ transition can be obtained analytically,

$$P_{\uparrow\to\downarrow}(\infty) = 1 - \exp\left[\frac{-\pi(\Delta^2+\gamma^2)}{2\hbar\nu}\right] \tag{6}$$

According to Eq. (6), the long-time limit of the transition probability, $P_{\uparrow\to\downarrow}(\infty)$, depends on the values of Δ and γ. Obviously, the time evolution of the LZ transition probabilities in Fig. 3 lead to asymptotic values in accordance with Eq. (6).

The purpose of our project is to predict the evolution of the LZ transition probability by using the model parameters Δ and γ. As a convenient check of the performance of the NNs, the predicted LZ transition probability in the long-time limit should be in accordance with the analytical expression of Eq. (6).

B. Outline of our NN approach

As shown in Fig. 4, our overall NN approach in this work can be regarded as a combination of one GAN, two CNNs, and multiple BPNNs. Based on the diagram in Fig. 4, we briefly describe the overall procedure. The first step is to use the multiple Davydov D_2 Ansatz to calculate multiple sets of LZ transition probability data as training data by varying the tunneling strength parameter Δ and the off-diagonal coupling strength parameter γ within certain ranges. This is to establish the training data, which is the part represented by the "Real data" in Fig. 4. The GAN uses random noise and "Real data" to generate a set of counterfeit data, which include all time series ($-20 < t < 40$, consisting of 6001 data points) and are different from any set of training data while retaining structural similarity. In the second step, the training data are divided into three time series, i.e., $-20 < t < 0$, $0 < t < 10$ and $10 < t < 40$. The time series of $-20 < t < 0$ is irrelevant for our purpose, as within this early time period the LZ transition does not occur. The second time series is controlled by the parameter Δ ($0 < t < 10$, about 1000 data points), and the third, mainly by the parameter γ ($10 < t < 40$). These two time series could be used as the training data for the two CNNs, which are to be trained separately to identify

the two parameters. As the operations of the two CNNs are similar, we only describe the more complex one, that is, the time series controlled by the parameter γ. After that, the trained CNNs are adopted to identify the corresponding part of the counterfeit data. Because the high-precision calculation method is time consuming, we have chosen to use low-precision data generated by the multiple Davydov D_2 Ansatz with multiplicity of $M = 4$ for training first, which is bound to result in certain errors in the prediction. Therefore, the third step is to divide the counterfeit data into multiple parts, and then use the BPNN to establish a mapping relationship for each part to reduce errors.

First of all, let us define several frequently used variable names. "Training data" is calculated by the multiple Davydov D_2 Ansatz with $M = 4$, Δ raging from 0.1 to 1.0, and γ ranging from 0.1 to 1.2. "Target data" is the learning target of the output data during NN training. Different NNs have different sets of target data. For example, in GAN and BPNN, the "target data" represents the training data, while in CNN the "target data" is the parameters Δ or γ used to calculate the training data. "Validation data" is calculated by the multiple Davydov D_2 Ansatz with $M = 10$, $\Delta = 0.45\sqrt{\nu/\hbar}$, and γ ranging from $1.3\sqrt{\nu/\hbar}$ to $2.0\sqrt{\nu/\hbar}$. "Analytical value" is the LZ transition probability in the long-time limit given by Eq. (6). "Convergent data" is the average value of training data over the time period of $15 < t < 40$, which approximates the LZ transition probability calculated by training data in the long-time limit. Relevant details are elaborated in Sec. III.C.

C. Artificial NNs used in our study

The core of artificial intelligence is machine learning, which has representation learning and deep learning as its center. The purpose of deep learning is to discover hidden hierarchical constructs with probability distributions over a variety of data. In deep learning, an entire probability distribution of the data set is generated explicitly, such as density estimation, or implicitly, such as synthesis or de-noising. Deep learning can be roughly classified into unsupervised and supervised learning algorithms: the former usually refers to the procedure of training a data set with many features and learning useful structural properties on these data, while other types of unsupervised learning, such as clustering, divide a data set into sets of similar samples; the latter usually refers to the procedure of training a data set with a target and learning how to classify samples into categories based on measurement outcomes.

Supervised learning implies that a known goal is provided to the system under training, and the system strives to achieve the given goal. In unsupervised learning, however, there is absence of goals, and the algorithm must learn to understand the data without any guidance. Traditionally, regression, classification, and structured output problems are referred to as supervised learning, and density estimation as unsupervised learning. In this project, GAN is adopted as unsupervised learning algorithms, while CNN and BPNN, supervised learning algorithms.

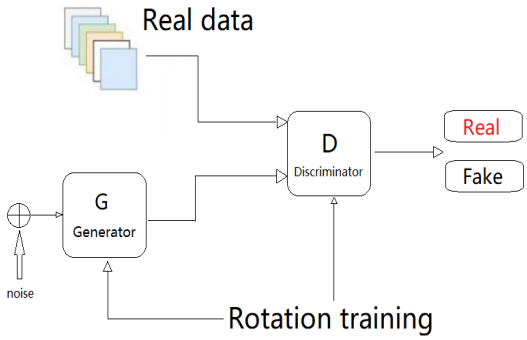

Figure 5: A flow chart of GAN. G is the generator, and D is the discriminator. The goal of G is to use a set of random noise as the initial data to generate a set of related data as the input of the discriminator, which the goal of D is to compare the target data with the data generated by G to determine whether the generated data is the target data.

1. *The GAN Model*

The output scale of an artificial NN is in general less than or equal to the input scale, which coincides with the law of information dissemination. A large amount of information can uniquely lead to a conclusion. Conversely, the inference does not hold. Therefore, in order to predict the occurrence probability of LZ transitions with a small number of parameters, the first thing to do is to train a NN to generate a set of counterfeit data with similar characteristics by involving random noise.

A flow chart of GAN is shown in Fig. 5. GAN consists of two parts, one of which is called the generator (G), and the other is called the discriminator (D) [36]. The goal of the generator G is to use a set of random noise as the initial data to generate a set of related data as the input of the discriminator. Then, according to the feedback output by the discriminator, it could continuously modify itself to achieve the ultimate goal of generating counterfeit data. Another input set of the discriminator is the training data, also called the target data. The goal of the discriminator D is to compare the target data with the data generated by G to determine whether the generated data is the target data. As the amount of training increases, the generated data could inevitably look more similar to the target data, with the discriminator D more capable in judging whether the generated data is the target data. Details on the GAN loss function can be found

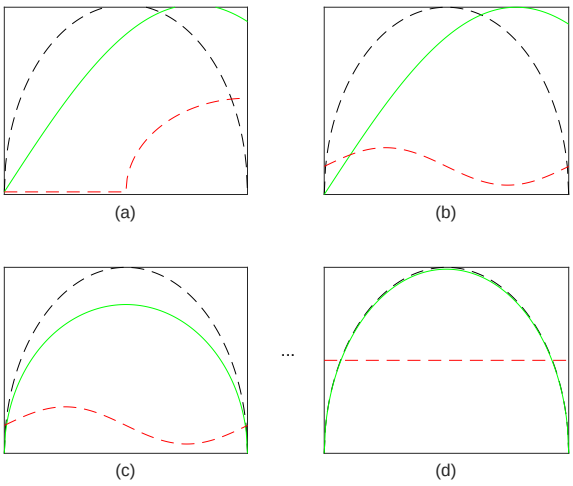

Figure 6: GANs are trained by updating the discriminative distribution (D, ochre dashed line) so that it discriminates between samples from the data generating distribution (black dashed line) and those of generative distribution (G, green solid line), demonstrating the competition between G and D. (a) before training, (b) training D, (c) training G, (d) after training

in Appendix B.

As these processes are repeated, the generator and the discriminator compete with each other, resulting in an improved performance of the two. Eventually, a relative balance is reached, that is, the probability that the discriminator thinks that the generated data is true data reaches 50%, then it can be considered that GAN has been fully trained. The GAN training process are illustrated in Fig. 6, where the black dashed line denotes the training (target) data, the green line, the counterfeit data generated by the generator, and the ochre dashed line, the probability that the discriminator considers the data to be true.

One can see the performance is not good for both the generator or the discriminator in Fig. 6(a), as the weights of G and D in the initial state are set randomly. I follows that the discriminator adjusts its weights and yields a new judgment, as shown in Fig. 6(b). The discriminator decides that the data generated by generator is not the training data, and based on the feedback, the generator adjusts its weights as exhibited in Fig. 6(c). Repeated adjustments and competition of the two finally yields a balance as shown in Fig. 6(d). The discriminator now decides that the probability that the generated data is the training data is 50%, and the probability that it is counterfeit data is 50%. In other words, GAN has been sufficiently trained.

2. The CNN Model

Generally speaking, CNN is used for recognition or classification in a supervised learning framework. Schematics of a CNN is shown in Fig. 7. The input layer processes the trained, two-dimensional image data as matrix data, and extracts the feature values of the training data in the convolution layer. In the pooling layer, the dimensionality of the feature values of the training data is reduced, thus the amount of calculations is also reduced. After the process of convolutional pooling, the scale of the two-dimensional image data is reduced to a level that can be easily calculated by a computer in the full connected layer.

In order to prevent overfitting, a "dropout" operation is usually performed in the fully connected layer according to the scale of the data. Depending on the dropout rate, which is in the range from 0.3 to 0.5, some data points could be "lost", which means that usually 30% to 50% of data does not participate in subsequent predictions. Only the remaining 50% to 70% of the data are passed to the output layer.

In order to prevent the input of each layer from being a linear function of the output of the previous layer, the data transfer between layers uses the activation function to perform a nonlinear mapping, and the general activation function is selected to be the ReLU function.

3. The BPNN Model

BPNN is one of the basic NNs in deep learning with back propagation, providing the network the necessary feedback to correct itself. Many neural networks, such as CNN and the recurrent neural network(RNN), use similar algorithms. A schematic diagram of the BPNN model is shown in the Fig. 8. The working principle of BPNN is as follows. First, the input layer passes the training data to the hidden layer, then the hidden layer processes the data, and passes it to the output layer. The BP algorithm is used to adjust the weight between layers until the difference between the target and the data from the output layer falls below an acceptable range.

The BPNN is usually used in function approximation or mode recognition. Function approximation is a method that uses the input vector and the corresponding output vector to train a network to approximate a given function, while mode recognition is a method that uses a pending output vector to associate it with the input vector. In this work, BPNN is used for function approximation.

III. RESULTS AND DISCUSSION

The training data set is composed of time series of LZ transition probabilities calculated via the multiple Davydov D_2 Ansatz in a time-dependent variation procedure

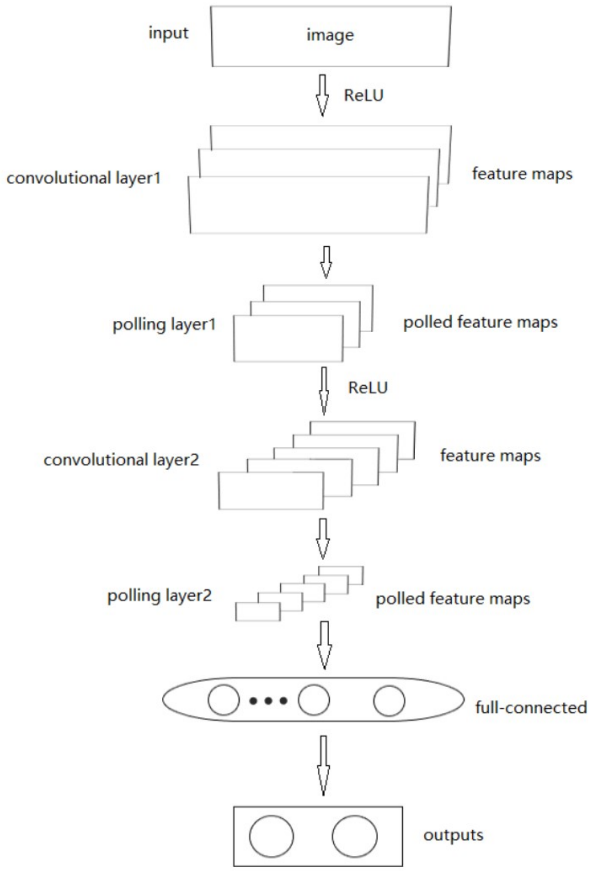

Figure 7: A schematic diagram of CNN. The purpose of CNN is to extract features from a certain model, and then classify, identify, predict or make decisions based on the features. Feature extraction is implemented by the convolutional layer and the pooling layer. In the convolutional layer, the data is divided into blocks and inner-producted with the convolution kernel one by one. The output is feature maps. In the pooling layer, the dimensionality of the feature maps needs to be reduced, reducing the complexity of data operations while retaining the basic information of the data. After that, feature maps are dropped out in the fully connected layer. In order to prevent data from overfitting, different activation functions are selected according to different purposes.

[34, 35], and each data set is determined by a parameter pair of γ and Δ. The value of γ ranges from 0.1 to 1.2, and that of Δ, from 0.1 to 1.0, both in increments of 0.1. The strength of intrinsic interactions between the diabatic states, Δ, is in general less than or equal to the environment-system coupling strength γ, in the pairing of Δ and γ. A total of 75 data sets are established for the training.

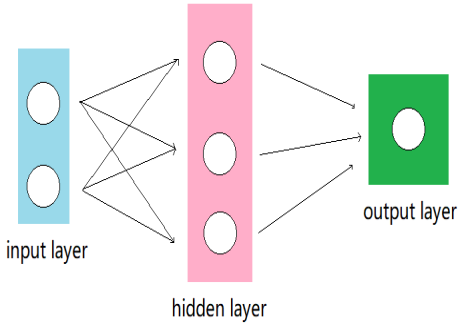

Figure 8: A schematic diagram of BPNN. On the left hand side is the input layer, where the input training data is stored, and the middle pink column represents the hidden layer. There can be multiple hidden layers, where the training data is nonlinearly transformed. On the right hand side is the output layer, where the data is finally exported.

Table I: Simulation outcomes for seven different cases. "training steps" labels the number of iterations, Lr(G) is the learning rate of G, and Lr(D) isthe learning rate of D. The convergence time, "time (h)", is given in units of hours. MSE is used to evaluate prediction results.

Case	1	2	3	4	5	6	7
training steps	2000	5000	10000	40000	40000	40000	100000
Lr(G)	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Lr(D)	0.1	0.1	0.1	0.1	0.12	0.15	0.1
time (h)	0.33	1.25	2.5	9.0	9.2	9.3	18.2
MSE(%)	11.43	10.48	5.71	1.46	0.23	0.79	2.43

A. Results from GAN

Due to instabilities in the GAN algorithm, it is difficult to achieve the Nash equilibrium as the cut-off condition of the network. Therefore, cut-off conditions are determined by training steps and the convergence speed of GAN depends on the training steps and the learning rate.

To ensure accuracy and to achieve a higher convergence speed, we carry out a series of test comparisons. As shown in Table. I, "training steps" labels the number of iterations, Lr(G) is the learning rate of G, and Lr(D) isthe learning rate of D. The convergence time, "time (h)", is given in units of hours; the smaller the convergence time it is, the higher the convergence speed. An index for evaluating the pros and cons of GAN results is the Mean Square Error (MSE), given by

$$\mathrm{MSE} = \sqrt{\frac{\sum_{i=1}^{n}(x_t - x_o)^2}{n}} \tag{7}$$

where x_t is the target (training) data and x_o is the output (counterfeit) data generated by GAN. The smaller the MSE is, the more accurate the results are.

As exhibited in Table I, more training steps do not always lead to better predictions. For example, Case 7 ($100,000$ training steps) has a larger MSE than Cases 4 to 6 ($40,000$ training steps), and Cases 1 to 3 ($2,000, 5,000, 10,000$ training steps, respectively) has a smaller MSE than Cases 4 to 6 ($40,000$ training steps).

In this work, 40,000 training steps are chosen. To increase the fidelity of the generated data, it is necessary to improve the discriminator performance. A better choice is Lr(G) = 0.1 and Lr(D) = 0.12. The final data size generated by GAN can be arbitrarily specified. In this work, 10 data sets are generated, one of which is shown in Fig. 9 (a). As the number of training steps reaches 40,000, the set of generated counterfeit data has already possessed the main characteristics of the LZ transition process. A comparison between the generated data set (blue) and the training data set (ochre) is plotted in Fig. 9 (b), demonstrating that the generated data set is not a simple copy of the training set. In particular, the counterfeit curve before $t = 0$ differs significantly from all training curves. As data before $t = 0$ do not affect the prediction of subsequent results, they are kept as a label of the counterfeit data.

Taking as input a set of random noise, and using the LZ transition probability and the image of its time evolution obtained by time-dependent variation as the training data set, the GAN has been fully trained to this point, producing a set of counterfeit images as its output that resembles the training set. Next, CNNs are adopted to unveil the information embedded in the set of counterfeit data.

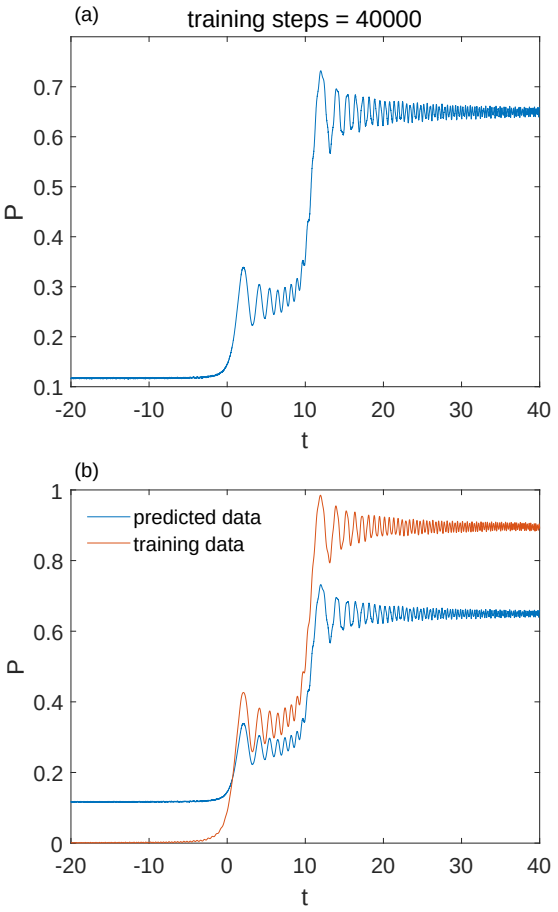

Figure 9: GAN training process display. (a) Generate data. (b) Comparison of generated data and random raw training data.

B. Results from CNN

As the parameters Δ and γ control the probability of LZ transitions, each group of time-dependent LZ transition probabilities contains the information on parameters Δ and γ, which is identified by the CNNs.

Before training the CNNs, a portion of the data can be processed artificially. According to Fig. 3, changes in parameter Δ affect the probability of the LZ transition $P_{\uparrow\to\downarrow}(t > 0)$, while changes in parameter γ, $P_{\uparrow\to\downarrow}(t > 10)$. The initial LZ transition occurs according to the condition $\Delta \leq \gamma$, inferring that for $0 < t < 10$, the probability of the LZ transition is mainly determined by Δ, and for $t > 10$, jointly determined by Δ and γ, but the influence of γ is dominant. Therefore, the data are first divided into three parts. The part dominated by Δ is called the first sequence (for $0 < t < 10$), and the part dominated by γ is called the second sequence (for $10 < t < 40$). The rest is temporarily put aside. The training data are usually selected from the first sequence or the second sequence, so the size of the training data is greatly reduced. The number of data points in the training data input is only about 1000∼2000. To save CPU time, a simplified one-dimensional CNN is utilized to extract the information on the control parameters. Since the data size is relatively small, there is no need to perform dimensionality reduction operations on the convolutional feature data, rendering unnecessary the pooling layer in CNN, and resulting in a simplified CNN structure shown in Fig. 10.

A total of two CNNs are trained here, one to identify Δ, and the other to identify γ. Setting the convolution kernel is a $1 \times n$ matrix, and each element in the matrix is $1/n$ with $n = 500$. The fully connected layer consists of two layers, each with 5 neurons. The activation function between input layer and convolution layer is ReLU, the activation function between other layers is Tanh, the optimizer is Gradient Descent, and the loss function is MSE.

Similar to the GAN, the training data required to train the CNN is also calculated by the multiple Davydov D_2 method. The time series data consist of 6001 data points. If the trained network is to recognize Δ (γ), the training data should be the data points from 2001 to 3000 (from 5001 to 6000). In this section, we mainly discuss the

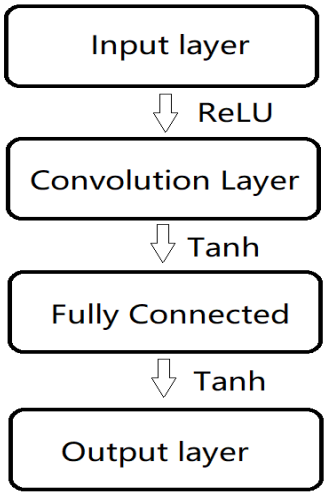

Figure 10: Simplified CNN structure. The activation functions between input layer and convolutional layer is ReLU, it is to remove all non-zero values in the training data. Then training data is convolved with a large convolution kernel in the convolutional layer, after which the training data is calculated in the fully connected layer. The activation functions of the remaining layers are Tanh.

Table II: Eight sets of data randomly selected from the test data. CNN is used to identify parameter information of test data. Label-γ and Label-Δ are true parameters of the selected data, and output-γ and output-Δ are outputs of CNN.

	1	2	3	4	5	6	7	8
Label-γ	1.0	0.6	0.7	0.5	0.9	0.7	0.7	0.4
Label-Δ	0.5	0.3	0.5	0.2	0.7	0.1	0.3	0.1
output-γ	1.05	0.62	0.68	0.51	0.87	0.68	0.68	0.42
output-Δ	0.52	0.31	0.48	0.21	0.67	0.10	0.29	0.11

CNN of identifying γ. Finally, the two networks are connected in series, and the original training data obtained by the multiple Davydov D_2 with $M = 4$ are divided into training data and test data according to the ratio of 8 : 2. The results of the sampling display are shown in Table II, and the accuracy rate is found to reach almost 95%.

From Fig. 11, it is found that the generated random data are recognized by CNN as $\Delta = 0.45\sqrt{\nu/\hbar}$, and $\gamma = 0.7\sqrt{\nu/\hbar}$. From GAN, data points of the counterfeit data are generated, and from CNN, we obtain the tunneling strength parameter Δ and the off-diagonal coupling strength γ corresponding to the counterfeit data. According to the known parameter information and that recognized by CNN, the correspondence between the two can be easily obtained. Based on this, the counterfeit data generated by GAN can be used for the final prediction. Relevant details are elaborated in Sec. III.C.

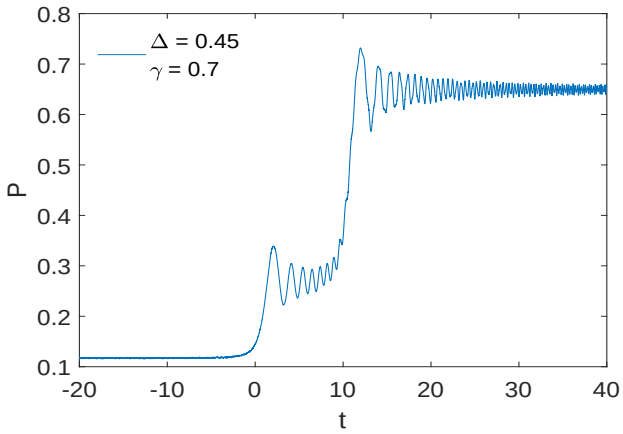

Figure 11: One of 10 sets of counterfeit data generated by GAN, after recognition by CNN. The parameter information of this data set is $\Delta = 0.45\sqrt{\nu/\hbar}$ and $\gamma = 0.7\sqrt{\nu/\hbar}$.

C. Results from BPNN

Similar to the process of training CNN to parameterize the network, the procedure of using BPNN for prediction is to divide the data into two time series, one of which is dominated by Δ ($0 < t < 10$), and the other, by γ ($10 < t < 40$). Due to the similarity of the two operations, in this work we show the prediction of the second sequence (dominated by γ in the interval of $10 < t < 40$) because it is more challenging.

According to the correspondence between the analytical value calculated using the given parameter and that using the GAN-recognized parameter, we can process the counterfeit data to approximate the data calculated by the multiple Davydov D_2 Ansatz with given parameters. Unfortunately, this is not the correct predicted result as it is found that the predicted LZ transition probability exceeds 1 during $10 < t < 15$ for a larger γ, which has no physical meaning. If the phrase "convergent value" of a time series is used to denote the average value of the last 2500 data points of the time series, then the convergent value of the predicted LZ transition probability deviates slightly from the analytic result Eq. (6). To handle these two prediction issues, we need to construct the mapping relationships with BPNN, a schematic diagram of which is shown in Fig. 12(a).

The second sequence is divided into two parts. The first part covers the time interval of $10 < t < 15$, and is referred to as the structural part (labeled by "str"). Five mapping relationships (i.e., structure mappings) are constructed between the input data (i.e., processed counterfeit data) and the target data (i.e., the training data) with BPNN to provide corrections to the predicted LZ transition probability. Here the input data are the coun-

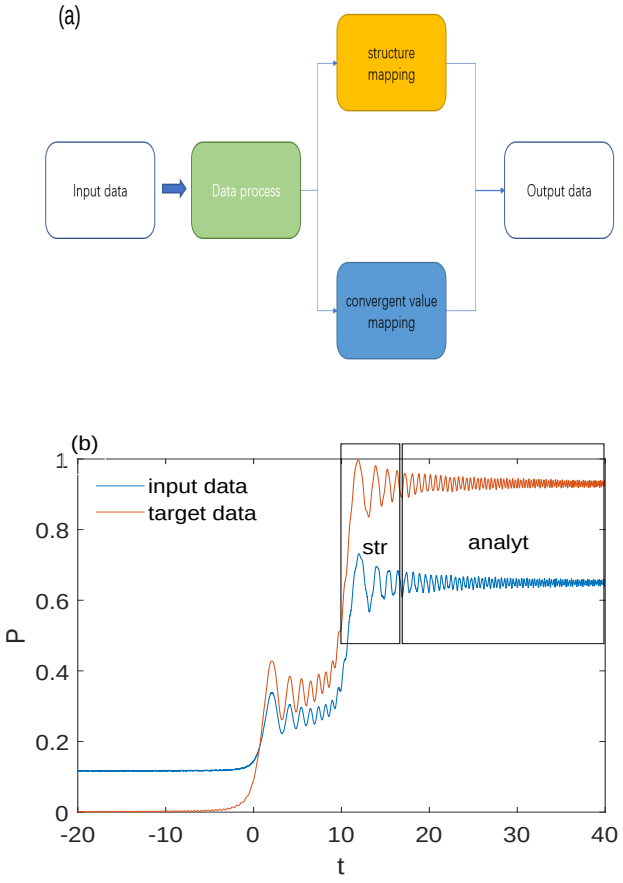

Figure 12: (a) Schematic diagram of our multiple BPNNs. The input data here is the counterfeit data generated by a trained GAN, "data process" denotes processing input data based on the correspondence between the analytical value calculated by the given parameter and that by the recognized parameter (through CNN), and "structure mapping" and convergent value mapping are two sub-neural networks, used to correct the predicted LZ transition probability. The output data here is the predicted data. (b) Division of the time series of $10 < t < 40$ into two portions: the square labelled as "analyt" represents the "convergent value part", and the squarelabelled as "str" represents the "structure part". The blue line denotes the counterfeit data generated by a trained GAN, serving as the input data in BPNN. The ochre line is a set of training data obtained from the multiple Davydov D_2 Ansatz, which is assumed as the target data.

terfeit data processed via the correspondence between the analytical value calculated with given parameters and that with recognized parameters (through CNN). Relevant details are explained in Sec. III C 1. The second part covers the time interval of $15 < t < 40$, and is referred to as the convergent value part labeled by "analyt". In the "analyt" part, we need only to construct one mapping relationship (i.e., convergent-value mapping) between the convergent value of input data and that of target data with BPNN. It is used to realize the correction of the convergent value of the counterfeit data generated by GAN, as shown in Fig. 12 (b)

1. *Low multiplicity*

The parameter information of the GAN counterfeit data can be obtained by the application of CNN. The target data of BPNN is provided by the multiple D_2 Ansatz with $M = 4$, and the GAN counterfeit data serves as the input data of BPNN. In order to ensure a higher accuracy of the predicted data, this mapping relationship between counterfeit data and target data should be established at all points in the time series, which is time consuming. However, as mentioned earlier, the time interval of $-20 < t < 0$ is not needed for predicting, while the time interval of $0 < t < 10$ can also be neglected because of extremely weak γ dependence. Only the rest 3000 data points from the interval of interval of $10 < t < 40$ in the time series will be used to construct the mapping relationships with BPNN.

After traversing the remaining 3000 data points, each corresponding to a point in time, it is found that the average value of the last 2500 data points in the time series has nearly the same convergent value, calculated according to the target data is different from the final probability of Eq. (6), as shown in Fig. 13(a). This discrepancy is due to adopting a low multiplicity for the Davydov D_2 state during the variational simulation in the interest of efficiency. Therefore, we use NNs to learn a mapping relationship (i.e., convergent value mapping) between the analytical value and the convergent value. The training results are compared with Eq. (6) in Fig. 13(b). It is found that the NNs have achieved good agreement between the analytical value of Eq. (6) and the convergent value. Details on the BPNN structure mapping can be found in Appendix C.

Thus the NNs need to totally train six kinds of mapping relations to improve the training efficiency. The number of hidden layers is set to 1, and the number of neurons in the hidden layer, 5. The learning function is trainlm, the learning rate is 0.0001, and the maximum number of learning times is 1000.

After BPNN training, the time evolution of the LZ transition possibility with $\Delta = 0.45\sqrt{\nu/\hbar}$ and $\gamma = 1.3\sqrt{\nu/\hbar}$ is predicted. The validation data is obtained by the multiple D_2 Ansatz with $M = 4$, $\omega = 10\sqrt{\nu/\hbar}$, $\Delta = 0.45\sqrt{\nu/\hbar}$, and $\gamma = 1.3\sqrt{\nu/\hbar}$. The overall comparison diagram is exhibited in Fig. 13 (c). The MSE between the predicted data and the validation data for $10 < t < 40$ in the time series is less than 0.1%, which is acceptable.

2. *High multiplicity*

If the multiplicity of our Ansatz is sufficiently large, e.g., $M = 10$, the numerically convergent LZ transition

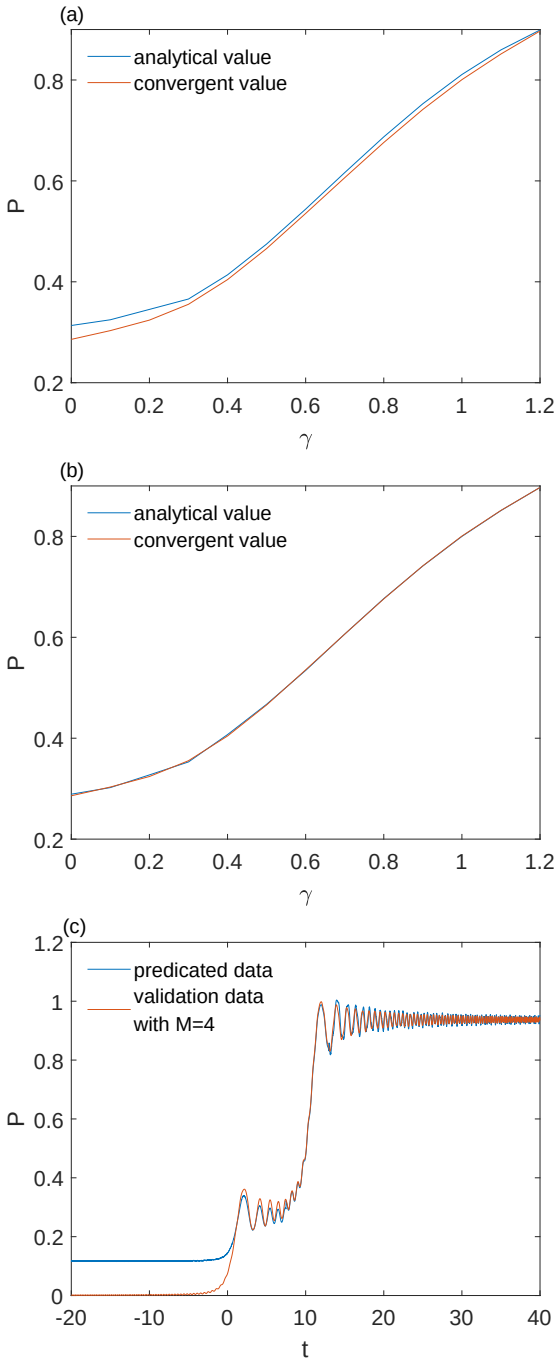

Figure 13: The BPNN training process. Comparison between the convergent value of training data and the final probability calculated by Eq.(6). (a) Before training. (b) After training. (c) Comparison of the final predicted data with the validation data. The blue line is the predicted data, and the ochre line is the validation data calculated by the multiple D_2 Ansatz with $M = 4$, $\omega = 10\sqrt{\nu/\hbar}$, $\Delta = 0.45\sqrt{\nu/\hbar}$, $\gamma = 1.3\sqrt{\nu/\hbar}$.

Table III: After training, the values of MSE between the predicted data and the real data calculated by the multiple Davydov D_2 Ansatz with $M = 10$.

γ	1.4	1.5	1.6	1.7	1.8	1.9	2.0
MSE(%)	0.34	0.39	0.42	0.46	0.24	0.54	0.56

probability, also known as the training data, agrees in the long-time limit with the analytical expression of Eq. (6), as shown in the Fig. 14(a). Therefore, it is only necessary to learn the "structural part", but not the "convergent value part", as our algorithm can directly predict the analytical value at long times. As a result, only 5 BPNNs are needed to complete the task.

Fig. 14(b) displays a comparison between predicted data with different multiplicities for the case of $\gamma = 1.3\sqrt{\nu/\hbar}$ and $\Delta = 0.45\sqrt{\nu/\hbar}$. The green (blue) line denotes the predicted data from a training data set with multiplicity $M = 10$ (4). The yellow dashed line is the steady-state value given by Eq. (6). As demonstrated in Fig. 14(b), predictions based the training data set with $M = 10$ have achieved a relatively higher accuracy in the long time limit.

In order to test the accuracy of our method, predicted data for $\gamma = 1.6$ and $\gamma = 1.8$ are shown in Figs. 15(a) and (b), respectively. Here the blue (ochre) line represents the predicted (validation) data. Overall, good agreement is achieved between the predicted data and the validation data calculated by the multiple D_2 Ansatz with $M = 10$. Furthermore, calculated MSE for 7 values of γ are listed in Table III, where the maximum MSE is found to be below 0.6%. Thus it can be directly inferred that the prediction results are reliable.

It is worth mentioning that only the time series for $10 < t < 40$ is shown in this work. But the time series for $0 < t < 10$ can also be predicted by a similar procedure, provided that the formula for predicting the final probability of the LZ transition used should be changed to

$$P(\infty) = 1 - \exp\left[\frac{-\pi\Delta^2}{2\hbar|\nu|}\right] \tag{8}$$

D. Numerical Efficiency

To sum up, a total of 13 NNs, including one GAN, two CNNs, and ten BPNNs, have been utilized in this deep-learning approach. Using training data calculated by the multiple Davydov D_2 Ansatz with the high multiplicity of $M = 10$, it takes about 188 hours to train the NNs and make predictions. To save CPU time, we choose instead as the training data those obtained by the multiple Davydov D_2 Ansatz with the low multiplicity of $M = 4$. In doing so, the training time of the NNs is shortened to 20 hours with the efficiency increased by 940%.

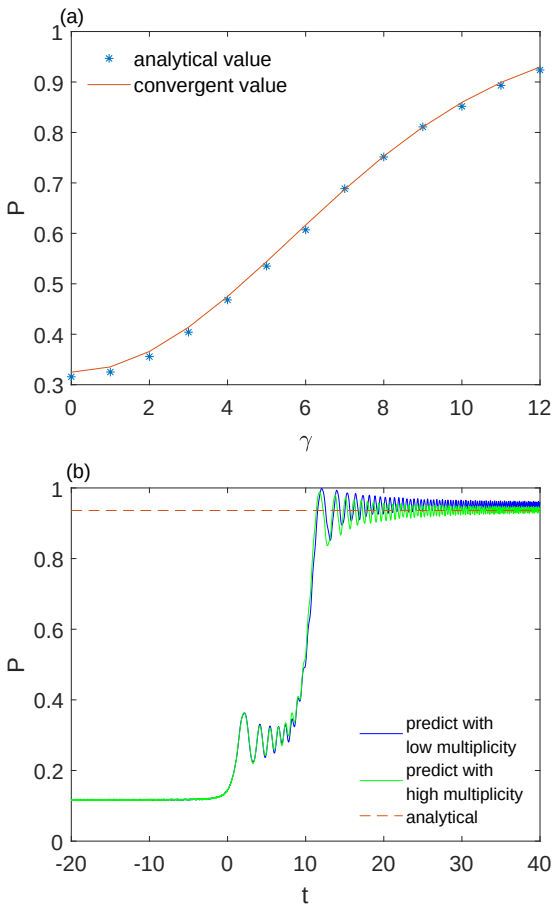

Figure 14: (a) Comparison between the training computed data set and the long-time transition probability of Eq. (6). The blue asterisk indicates the converged transition probability at long times from the multiple Davydov D_2 Ansatz, and the ochre line is calculated by Eq. (6). The two are in good agreement. (b) Comparison of predicted data using training data with different multiplicities for the time period of $-20 < t < 40$. The off-diagonal coupling strength γ is set to 1.3. The green line is the predicted data from a training data set with multiplicity of 10, and the blue line denotes the predicted data from a training data set with multiplicity of 4. The ochre dashed line indicates the steady-state value of Eq. (6). It is found that predictions based the training data set with $M = 10$ have achieved a relatively high accuracy at long times.

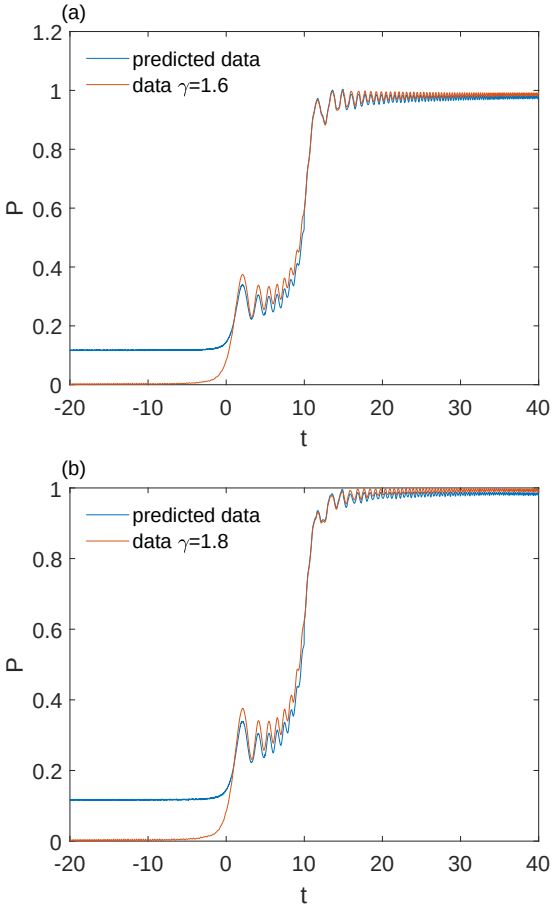

Figure 15: Comparison between the predicted data (blue line) and the validation data (ochre) calculated by the multiple D_2 Ansatz with $M = 4$, $\omega = 10\sqrt{\nu/\hbar}$, $\Delta = 0.45\sqrt{\nu/\hbar}$. (a) $\gamma = 1.6\sqrt{\nu/\hbar}$, (b) $\gamma = 1.8\sqrt{\nu/\hbar}$.

Table IV: Hardware specifications

OS	Windows 10
CPU	Intel(R) Core(TM) i7-8750H CPU
GPU	NVIDIA GeForce GTX 1050 Ti
Memory	8GB

The overall training time includes the time to train the NNs and the time to prepare the training data, with 46% of the time spent on the former. After the training is completed, the NN parameters are saved to facilitate the subsequent predication. So there is no need for our NNs to be retrained, or for the training data to be recalculated. Thus, regardless of the time partition between NNs training and training data preparation, it only takes 12.78 seconds to make the prediction using NNs. In comparison, it usually takes about 20 to 30 minutes for the multiple Davydov D_2 Ansatz with the high multiplicity of $M = 10$ to generate a set of converged data. The hardware configurations used in this work is shown in Table V. Compared to the high-precision dynamics data from the multiple Davydov D_2 Ansatz with a multiplicity of $M = 10$, the error rate of our more efficient NN procedure falls below 0.6 %.

IV. CONCLUSIONS

Previously, the nonlinear autoregressive NNs have been successfully employed by some of us to predict the nonadiabatic dynamics of a paradigmatic model with only one stage of the LZ transition [37]. But success was

elusive in predicting more complicated scenarios, where there are two or more transition stages. To overcome those shortcomings, we construct a procedure of three NN algorithms to jointly predict these more complex dynamical behaviors in the LZ transitions.

In particular, the first step of our procedure is to use random noise and time-dependent LZ transition probabilities calculated by the multiple D_2 Ansatz to generate a set of counterfeit data. The second step is to use two CNNs to identify the parameter information that is embedded in the generated counterfeit data. The third step is to establish two types of mapping relationships with BPNN: one is constructed between the final probability of Eq. (6) and the long-time asymptotic value of the training data obtained from the multiple Davydov D_2 Ansatz, while the other is established between the "structural part" of the training data and that of the counterfeit data generated by the GAN. Combining these three steps, the final output data can perfectly predict the evolution of the LZ transition probability for varying values of γ. Specifically, our approach has the potential to overcome accumulative numerical errors that are generated by the traditional method over for a long computation time.

With its demonstrated high efficiency, the machine learning approach as laid out in this work can be applied to many computational tasks in complex, many-body systems, such as noise characterization, parameter estimation, feedback, optimization of quantum control, in order to simplify and speed up resources-demanding calculations.

Acknowledgments

The authors gratefully acknowledge the support of the Singapore Ministry of Education Academic Research Fund (Grant Nos. 2018-T1-002-175 and 2020-T1-002-075)). K. Sun would also like to thank the Natural Science Foundation of Zhejiang Province (Grant No. LY18A040005) for partial support. L.L. Gao acknowledges the support of the Graduate Scientific Research Foundation of Hangzhou Dianzi University.

V. DATA AVAILABILITY

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

Appendix A: DERIVATION OF THE FINAL PROBABILITY OF LZ TRANSITION

For a nondegenerate initial state $|a\rangle$, the exact nonadiabatic Landau-Zener transition probability is given by Ref. [38]:

$$P_{a\to a} = \exp(-\frac{2\langle a|X^2|a\rangle}{\hbar\nu}). \tag{A1}$$

If at $t=-\infty$, the system starts in a state $|\uparrow k_+\rangle$ for which the diabatic energy is non-degenerate, then the following transition probabilities at $t=\infty$ are exact:

$$P_{\uparrow k_+\to\uparrow k'_+} = \begin{cases} \exp(-\frac{2\pi\langle\uparrow k_+|\nu^2|\uparrow k_+\rangle}{\hbar\nu}), & k'_+ = k_+ \\ 0, & k'_+ > k_+. \end{cases} \tag{A2}$$

By tracing out the environment, and performing the sum over k'_+ , we find:

$$P_{\uparrow\to\uparrow} = \exp(-\frac{\pi W^2}{2\hbar\nu}) = 1 - P_{\uparrow\to\downarrow}. \tag{A3}$$

with the ground-state expectation value:

$$W^2 = 4\langle\uparrow 0_+ \mid \nu^2 \mid\uparrow 0_+\rangle, \tag{A4}$$

where the off-diagonal part in Hamiltonian Eq. 1 reads:

$$\nu = \frac{\Delta}{2}\hat{\sigma_x} + \frac{\gamma}{2}\sigma_x(\hat{b} + \hat{b^+}), \tag{A5}$$

and will be called the bit-flip interaction, then we have:

$$\nu^2 = \frac{\Delta^2}{4} + \frac{\gamma^2}{4}(\hat{b}\hat{b} + \hat{b^+}\hat{b} + \hat{b}\hat{b^+} + \hat{b^+}\hat{b^+}). \tag{A6}$$

So,

$$W^2 = 4\langle\uparrow 0^+ \mid \frac{\Delta^2}{4} + \frac{\gamma^2}{4}(\hat{b}\hat{b} + \hat{b^+}\hat{b} + \hat{b}\hat{b^+} + \hat{b^+}\hat{b^+}) \mid\uparrow 0^+\rangle. \tag{A7}$$

Finally it can be cast into a simple expression:

$$W^2 = \Delta^2 + \gamma^2 \tag{A8}$$

Thus,

$$P_{\uparrow\to\uparrow} = \exp\left[-\frac{\pi(\Delta^2+\gamma^2)}{2\hbar\nu}\right] = 1 - P_{\uparrow\to\downarrow}. \tag{A9}$$

For more details on the derivation of Eq. 6, the readers are referred to Ref. [38].

Appendix B: LOSS FUNCTION OF GAN

Next, we explain the loss function of GAN:

$$\begin{aligned}\min_G \max_D V(G,D) = {} & E_{\mathrm{P_{data(x)}}}[\log(D(x))] \\ & + E_{\mathrm{P_z(z)}}[\log(1 - D(G(z)))]\end{aligned} \tag{B1}$$

As shown in Eq. (B1), where $D(x)$ denotes the output of discriminator D, x denotes the target data. And $G(z)$ denotes the output of generator G, z denotes the random noise. $V(G,D)$ is the loss function of GAN [36]. In order

to minimize the value of $V(G,D)$, Both $E_{\mathrm{P_{data}(x)}}$ and $E_{\mathrm{P_z(z)}}$ must be the smallest. Thus, the actual goal of GAN is to get the largest $D(x)$ while keeping $G(z)$ as small as possible.

We know that the training method of GAN is that discriminator D and generator G are trained separately in turn, so this objective function is also optimized for the discriminator and generator, respectively. First, the discriminator is optimized. The expression is as follows:

$$\max_D V(G,D) = E_{\mathrm{P_{data}(x)}}[\log(D(x))] + E_{\mathrm{P_z(z)}}[\log(1-D(G(z)))] \tag{B2}$$

where $D(x)$ denotes the output of discriminator D, and the output of $D(x)$ is a scalar between 0 and 1. Obviously, we hope that the discriminant result is as close to 1 as possible, and z is random noise, $G(z)$ represents the generated data. In discriminator, we hope that the discriminator's discriminant result $D(G(z))$ is as close to 0 as possible, that is, to minimize the value of Eq. (B2).

$$\min_G V(G,D) = E_{\mathrm{P_z(z)}}[\log(1-D(G(z)))] \tag{B3}$$

After the discriminator optimization, we proceed to optimize the generator. Similarly, for the generator, the value of its loss function Eq. (B3) should be as small as possible. It only needs to make the result of $D(G(z))$ close to 1 as possible, and this is obviously the opposite of the goal of the discriminator. Therefore, the confrontation between G and D arise.

If we need to use GAN to generate data under specific parameters, we can change the loss function of GAN as follows:

$$\min_G \max_D V(G,D) = E_{\mathrm{P_{data}(x)}}[\log(D(x \mid y))] + E_{\mathrm{P_z(z)}}[\log(1-D(G(z \mid y)))] \tag{B4}$$

where y is the label (i.e., specified parameters). Eq. (B4) is the loss function of Conditional Generative Adversarial Net (CGAN). Unlike GAN, the input of CGAN is random noise and labels, and the output data is not random counterfeit data, but closely related to the input label. When the label y is different, the loss function of CGAN will also be different. Therefore, by changing the label, we can generate the counterfeit data that we need.

textbfThe counterfeit data obtained through CGAN is equivalent to a simple reproduction of the training data set, while our method can predict the data exceeding the parameter range of the training data set. Therefore we chose GAN over CGAN.

Appendix C: STRUCTURE MAPPING OF BPNN

Fig. 16 displays the prediction result if the NN only learns one mapping relationship, i.e., "convergent value mapping", but ignores the other mapping relationship ("structure mapping"). It is found that the predicted data differ substantially from the validation data and some values of the predicated data for $10 < t < 15$ even exceed 1, which is not physical. Therefore, the "structure mapping" is indispensable in the prediction making. In the "structure mapping", the data points in $10 < t < 15$ (500 data points) are adopted as the part of the learning process to avoid the aforementioned nonphysical outcome

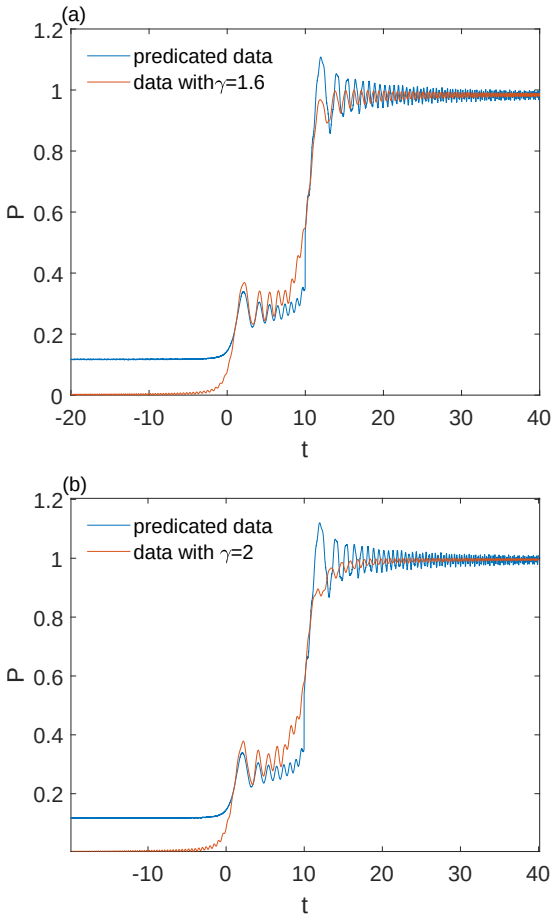

Figure 16: Comparison between the predicted data (blue line) and the validation data (ochre) calculated by the multiple D_2 Ansatz with $M = 4$, $\omega = 10\sqrt{\nu/\hbar}$, and $\Delta = 0.45\sqrt{\nu/\hbar}$ without the "structure mapping". (a) $\gamma = 1.6\sqrt{\nu/\hbar}$, (b) $\gamma = 2.0\sqrt{\nu/\hbar}$.

Table V: The number of established mapping relations versus the time for the NN result to converge and the MSE between the predicated data and the target data in the "structural part".

number	1	2	4	5	10	20
(MSE%)	2.3	1.6	1.1	0.8	0.5	0.3
time(min)	10.32	14.45	40.16	50.70	97.54	198.41

in the predicted LZ probability.

To ensure efficiency and accuracy simultaneously, we found it is appropriate to divide the target data into the "structural part" and the "convergent value part", and to train five corresponding mapping relationships in the "structural part", as shown in Table V. Similar to the "convergent value mapping", input data here is given by Eq. (6), the analytical expression of the LZ transition probability in the long-time limit. And target data is the average value of the corresponding training data. The number of hidden layers is set to 1, and the number of neurons in the hidden layer, 5. The learning function is trainlm, the learning rate is 0.0001, the maximum number of learning times is 1000, and validation check is 20.

[1] C. Zener, Proc. R. Soc. London. A **137**, 696 (1932).
[2] L. D. Landau, Phys. Z. Sowjetunion. **2**, 46 (1932).
[3] A. Thiel, J. Phys. G Nucl. Part. Phys. **16**, 867 (1990).
[4] R. J. Lipert, G. Bermudez, and S. D. Colson, J. Phys. Chem. **92**, 3801 (1988).
[5] W. Xie and W. Domcke, J. Chem. Phys. **147**, 184114 (2017).
[6] L. Zhu, A. Widom, and P. M. Champion, J. Chem. Phys. **107**, 2859 (1997).
[7] D. Bouwmeester, N. H. Dekker, F. E. v. Dorsselaer, C. A. Schrama, P. M. Visser, and J. P. Woerdman, Phys. Rev. A. **51**, 646 (1995).
[8] G. D. Fuchs, G. Burkard, P. V. Klimov, and D. D. Awschalom, Nat. Phys. **7**, 789 (2011).
[9] J. N. Onuchic and P. G. Wolynes, J. Phys. Chem. **92**, 6495 (1988).
[10] J. R. Petta, H. Lu, and A. C. Gossard, Science **327**, 669 (2010).
[11] J. Zhou, P. Huang, Q. Zhang, Z. Wang, T. Tan, X. Xu, F. Shi, X. Rong, S. Ashhab, and J. Du, Phys. Rev. Lett. **112**, 010503 (2014).
[12] G. Sun, X. Wen, M. Gong, D.-W. Zhang, Y. Yu, S.-L. Zhu, J. Chen, P. Wu, and S. Han, Sci. Rep. **5**, 8463 (2015).
[13] A. J. Olson, S.-J. Wang, R. J. Niffenegger, C.-H. Li, C. H. Greene, and Y. P. Chen, Phys. Rev. A **90**, 13616 (2014).
[14] W. D. Oliver, Y. Yu, J. C. Lee, K. K. Berggren, L. S. Levitov, and T. P. Orlando, Science **310**, 1653 (2005).
[15] T. Niemczyk, F. Deppe, H. Huebl, E. P. Menzel, F. Hocke, M. J. Schwarz, J. J. Garcia-Ripoll, D. Zueco, T. Hummer, E. Solano, A. Marx, and R. Gross, Nat. Phys. **6**, 772 (2010).
[16] I. Chiorescu, P. Bertet, K. Semba, Y. Nakamura, C. J. P. M. Harmans, and J. E. Mooij, Nature **431**, 159 (2004).
[17] A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, R.-S. Huang, J. Majer, S. Kumar, S. M. Girvin, and R. J. Schoelkopf, Nature **431**, 162 (2004).
[18] M. Wubs, K. Saito, S. Kohler, P. Hanggi, and Y. Kayanuma, Phys. Rev. Lett. **97**, 200404 (2006).
[19] J. Johansson, M. H. S. Amin, A. J. Berkley, P. Bunyk, V. Choi, R. Harris, M. W. Johnson, T. M. Lanting, S. Lloyd, and G. Rose, Phys. Rev. B. **80**, 12507 (2009).
[20] N. Zhou, L. Chen, Z. Huang, K. Sun, Y. Tanimura, Y. Zhao, J. Phys. Chem. A. **9**, 120 (2016).
[21] L. Wang, L. Chen, N. Zhou, Y. Zhao, J. Chem. Phys. **2**, 024101 (2016).
[22] G. Yang, N. Wang, T. Chen, K. Sun, Y. Zhao, J. Phys. Chem. C. **5**, 116 (2012).
[23] N. Wu, K. Sun, Z. Chang, Y. Zhao, J. Chem. Phys. **12**, 136 (2012).
[24] K. Sun, Y. Fujihashi, A. Ishizaki, Y. Zhao, J. Chem. Phys. **20**, 144 (2016).
[25] Z. Huang, Y. Zhao, Phys. Rev. A. **97**, 13803 (2018).
[26] Y. Hu, D. Yang, Z. Zhang, J. Phys. Conf. Ser. **1533**, 032093 (2020).
[27] A. Nada, H. Al-Baity, Sensors **20**,19 (2020).
[28] S. D. Ting, L. Pasquale, L. Peng, Brit. J. Ophth. **103**, 2 (2018).
[29] R. Socher, C. C. Lin, A. Y. Ng, C. Manning, Proc. 26th Int. Conf. Mach. Learn. (ICML 2011), **2**, 129-136 (2011).
[30] N. Jaitly, P. Nguyen, A. W. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition", in *Proc. Interspeech*, 2012.
[31] Y. Lecun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard. L. Jackel, Neur. Comp. **4**, 541-557 (2014).
[32] B. Yang, Z. Hui, H. Hu, X. Hu, L. Yu, IEEE Vis. Comm. Imag. Proc. **4** (2020).
[33] L. Wang, L. P. Chen, N. J. Zhou, Y. Zhao, J. Chem. Phys. **144**, 024101 (2016).
[34] Y. Zhao, B. Luo, Y. Zhang, and J. Ye, J. Chem. Phys. **137**, 084113 (2012).
[35] Y. Zhao, D. W. Brown, and K. Lindenberg, J. Chem. Phys. **107**, 3159 (1997).
[36] I. Goodfellow, J. PougetAbadie, M. Mirza, Adv. Neur. Info. Proc. Sys. **2672**, 2680 (2014).
[37] S. Bandyopadhyay, Z. K. Huang, K. W. Sun, Y. Zhao, Chem. Phys. **515**, 272-278 (2018).
[38] K. Saito, M. Wubs, S. Kohler, Y. Kayanuma, P. Hänggi, Phys. Rev. B **75**, 214308 (2007)