RESEARCH ARTICLE | MARCH 15 2023

Application of artificial neural networks for modeling of electronic excitation dynamics in 2D lattice: Direct and inverse problems

Special Collection: 2022 Machine Learning

Pranas Juknevicius ; Jevgenij Chmeliov ; Leonas Valkunas ; Andrius Gelzinis

AIP Advances 13, 035224 (2023)
https://doi.org/10.1063/5.0133711

Application of artificial neural networks for modeling of electronic excitation dynamics in 2D lattice: Direct and inverse problems

Cite as: AIP Advances 13, 035224 (2023); doi: 10.1063/5.0133711
Submitted: 5 November 2022 • Accepted: 16 February 2023 •
Published Online: 15 March 2023

Pranas Juknevicius,[1,2] Jevgenij Chmeliov,[1,2] Leonas Valkunas,[1,2] and Andrius Gelzinis[1,2,a)]

AFFILIATIONS

[1] Institute of Chemical Physics, Faculty of Physics, Vilnius University, Saulėtekio 9-III, 10222 Vilnius, Lithuania
[2] Department of Molecular Compound Physics, Center for Physical Sciences and Technology, Saulėtekio 3, 10257 Vilnius, Lithuania

[a)] **Author to whom correspondence should be addressed:** andrius.gelzinis@ff.vu.lt

ABSTRACT

Machine learning (ML) approaches are attracting wide interest in the chemical physics community since a trained ML system can predict numerical properties of various molecular systems with a small computational cost. In this work, we analyze the applicability of deep, sequential, and fully connected neural networks (NNs) to predict the excitation decay kinetics of a simple two-dimensional lattice model, which can be adapted to describe numerous real-life systems, such as aggregates of photosynthetic molecular complexes. After choosing a suitable loss function for NN training, we have achieved excellent accuracy for a direct problem—predictions of lattice excitation decay kinetics from the model parameter values. For an inverse problem—prediction of the model parameter values from the kinetics—we found that even though the kinetics obtained from estimated values differ from the actual ones, the values themselves are predicted with a reasonable accuracy. Finally, we discuss possibilities for applications of NNs for solving global optimization problems that are related to the need to fit experimental data using similar models.

I. INTRODUCTION

Machine learning (ML) techniques, in general, and artificial neural networks (NNs), in particular, are becoming of ever-increasing importance in most of the physical sciences.[1] In the field of chemical physics, they have been applied to quantum chemistry and atomistic molecular physics,[2–4] the calculation of Frenkel exciton parameters,[5] analysis of quantum dynamics of open molecular systems,[6–8] and calculation of multidimensional optical spectra.[9,10] Most of these applications correspond to regression type problems when a trained machine learning system has to predict one or several numerical values from numerical input. In such cases, deep NNs are of particular interest as they are able to learn deep and complicated relationships between the input and the output data.[11]

While the problems listed above are naturally important and rather general, researchers from the chemical physics community are nonetheless often faced with perhaps more mundane, but still difficult and time-consuming tasks. A significant amount of them involves solving an inverse problem—describing some physical phenomena with a model and then obtaining the values of the model parameters that bring the model predictions as close to the experimental values as possible. Thus, a global optimization problem has to be solved. This requires a considerable numerical effort as often good solutions can be consistently obtained only with the help of population-based methods, such as genetic algorithm,[12] particle swarm optimization,[13] or differential evolution (DE).[14] Thus, it is of interest to apply ML algorithms to such problems in order to reduce the required computational time.

Recently, we have fitted experimental time-resolved fluorescence data of aggregates of both major and minor light-harvesting complexes (LHCs) of higher plants (LHCII and CP29, respectively)[15–17] using a two-dimensional (2D) lattice model, where each site corresponded to one of the possible conformational states of an LHC, and excitation could either hop between them or decay

to the environment as heat or photon emission. By solving the resulting kinetic equations, the overall fluorescing population was obtained, which could then be related to the experimental fluorescence kinetics. The fitting involved application of the DE approach and many time-consuming calculations. It is probable that the analysis of experimental data from similar systems could also make use of such a model. Therefore, we are motivated to investigate whether NNs could be applied to speed up this kind of work.

For simplicity, in this work, we have considered a simplified version of the model used in Refs. 15–17—a square lattice, where each site could be in one of two different states. Please note, however, that the idea behind such a model is rather general and with some additional complexity, such as different lattice arrangements (e.g., hexagonal lattice), disordered (in terms of site energy or positions) lattice, different number of possible states, or Monte Carlo simulations rather than solving kinetics equations, this type of model could be used in studies of many completely distinct systems, from hopping of charge carriers in disordered organic materials,[18] exciton diffusion in perovskites,[19] fluorescence quenching in disordered 2D systems,[20] to energy migration in nano-engineered light-harvesting antenna arrays[21] or energy transfer and trapping in photosystems with different antenna sizes.[22]

It must be emphasized, however, that the application of ML algorithms can bear a significant computational overhead. The majority of it comes from two sources: obtaining the required training data and training the actual ML system. Additional complexity arises from the fact that ML algorithms often have metaparameters, the values of which have to be tuned for each specific application. Nonetheless, a trained ML system can usually perform calculations with a negligible numerical cost if compared to that required to obtain the training data. Thus, using ML algorithms can be extremely beneficial if a large number of such calculations need to be performed. The costs and benefits of ML systems have to be weighted for every application, while also keeping in mind that it is not always possible for them to reach the desired level of accuracy.[1]

In this work, we investigate the applicability of deep, sequential, fully connected NNs to the aforementioned lattice model. We analyze both the direct problem—predicting the kinetics of the fluorescing population from the values of model parameters—and the inverse problem—predicting the values of the model parameters from the kinetics curve. We find that it is crucial to select a suitable loss function in the training of NNs. Finally, we propose a strategy for the utilization of NNs for solving global optimization problems that arise from the need to fit experimental data with similar models.

II. METHODS

We seek to describe the electronic excitation dynamics in a 2D square lattice by modeling the outgoing fluorescence signal, which can be represented as the total fluorescing population. We assume that each site of the lattice can be in either a fluorescing (bright) or a quenched (dark) state. The excitation propagates in the lattice by hopping between two adjacent sites with the corresponding transfer rate, but it can also decay radiatively or non-radiatively with the corresponding relaxation rate. We use a large but finite lattice of size 10×10, similar to Ref. 15, together with five parameters to describe the model. The hopping rates between two bright sites, from the bright to dark site or from the dark to dark site, are chosen to be the same for simplicity and are denoted as k_{hop}. This assumption means that the free energy of the dark states is lower than those of bright states; thus, dark states act as shallow or deep excitation traps. The hopping rate from the dark site to the bright site is k_{BD}. Furthermore, there are two relaxation rates: k_{r} for the bright site and k_{trap} for the dark site. See Fig. 1 for an illustration of the lattice.

Real-life systems are usually heterogeneous. This means that, in experiments, an ensemble of systems is usually probed. The statistical properties of the 2D lattice considered in this work are characterized by the final parameter of the model, n_{D}, which is the average number of dark sites per lattice.

If we denote the probability that the ith site is excited as $P_i(t)$, then to obtain the total excitation decay kinetics of this system, we have to solve the system of Pauli master equations

$$\frac{\mathrm{d}}{\mathrm{d}t}P_i(t) = \sum_{j\neq i} k_{i\leftarrow j}P_j(t) - \sum_{j\neq i} k_{j\leftarrow i}P_i(t) - P_i(t) \times \begin{cases} k_{\mathrm{r}}, & \text{if } i \in \text{bright;} \\ k_{\mathrm{trap}}, & \text{if } i \in \text{dark.} \end{cases} \tag{1}$$

We assume that the initial excitation is distributed homogeneously across the whole lattice. This initial condition corresponds to the excitation of the lattice with an ultra-short laser pulse. Thus, our description considers single excitation per lattice, neglecting any non-linear effects.

The intensity of the fluorescence of our considered system would be proportional to the total population of the bright sites as we assume that the quenched sites fluoresce at other wavelengths if at all. Due to the statistical nature of this description, we calculate the averaged kinetics of 1000 realizations with different lattice arrangements and the actual number of dark complexes from the binomial

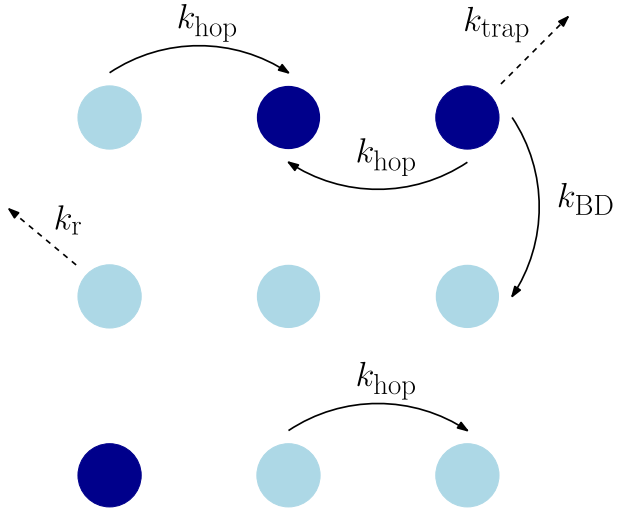


FIG. 1. Schematic picture of the 2D lattice considered in this work. Light blue circles represent bright states and dark blue circles represent dark states. Arrows represent possible hopping and relaxation rates.

distribution and the same hopping and relaxation rates. Thus, the averaged fluorescing population kinetics is then calculated as

$$P(t) = \left\langle \sum_{i \in \text{bright}} P_i(t) \right\rangle. \tag{2}$$

Many different systems could be described by the considered lattice model, depending on the actual values of the parameters. We are interested in the ability of NNs to model a variety of such systems. Thus, we consider many different sets of parameter values. For simplicity, we take the ranges of parameter values to be similar to previous modeling of real-life processes taking place in aggregates of light-harvesting complexes:[15] $k_{\text{hop}} \in (0.005\text{–}0.04)\ \text{ps}^{-1}$ and $k_{\text{BD}} \in (3.3 \cdot 10^{-5}\text{–}0.04)\ \text{ps}^{-1}$. k_{r} is fixed at $0.0002\ \text{ps}^{-1}$, corresponding to the decay time of 5 ns, which approximately represents many chlorophyll systems.[23] We also take $k_{\text{trap}} \in (0.0002\text{–}0.02)\ \text{ps}^{-1}$, which corresponds to an average relaxation lifetime of (50–5000) ps for the quenched states. Finally, the average number of dark states is between 0.5 and 15 per 100 lattice sites. All the parameter values are summarized in Table I. Note that all the parameter values were sampled from the uniform distribution of timescales, and then, rates were calculated as inverse lifetimes. Thus, it could occur that $k_{\text{hop}} < k_{\text{BD}}$, corresponding to situations when the free energy of the dark states is higher than those of bright states. As these situations could occur very rarely, such parameter value combinations were kept as they correspond to another regime that NNs should learn.

All calculations are done up to 15 ns, as this time is enough to cover the signal decay by 2–3 orders of magnitude. We obtain $P(t)$ with a 2 ps discretization time step, corresponding to 7500 data points in total.

Our goal in this work is to use the NNs to relate the values of the model parameters to the corresponding averaged fluorescence population kinetics. In the direct problem, the latter should be predicted by using the former, while in the inverse problem, it is the values of the model parameters that have to be predicted from the kinetics.

When considering the direct problem, the most straightforward approach would be to use the obtained kinetics as input data for the NN. On the other hand, the information content of the obtained kinetics is significantly smaller than the number of all the data points. Since kinetics is always a monotonically decaying function, it is convenient to first fit it with some analytical function with a small number of parameters. This corresponds to using a reduced dimensionality data description. In this way, we can predict the parameter values of the fitting function, rather than the averaged kinetics itself. It must be emphasized that choosing the fitting function is not trivial because ML algorithms have difficulties in predicting the fitted coefficients of certain functions. Therefore, we must find a fitting function that fits the data accurately and the chosen algorithm would be able to learn the coefficients of the function. We have used two functions for the fits of kinetics, either three terms of the exponential function,

$$f(t) = A\exp(Bt) + C\exp(Dt) + E\exp(Ft), \tag{3}$$

or two terms of stretched exponential functions,

$$f(t) = A\exp(Bt^C) + D\exp(Et^F), \tag{4}$$

$$f(t) = A\exp(Bt^C) + (1-A)\exp(Et^F). \tag{5}$$

Note that (3) and (4) fitting functions have six parameters, while (5) is a normalized version of (4) fitting function with five parameters. The rationale for using unnormalized ($f(0) \neq 1$) functions is that they might fit the overall behavior of the kinetics better, albeit with some differences at very early times. These functions can represent the non-exponential behavior of the kinetics $P(t)$ with a limited number of parameters. To fit the averaged kinetics, we used the DE method.[14]

In addition to NNs, we have tested many other ML algorithms for this task (Random forest regression, k-nearest neighbors algorithm, MultiOutputRegressor, LinearRegression, etc), but they were outperformed by the NN. This is illustrated for KNeighborsRegressor (realization of k-nearest neighbors algorithm in Python *scikit-learn* library) below in Sec. III. In the rest of this paper, we will, thus, focus on the results based on the NN.

TABLE I. Lattice parameters and their value ranges considered in this work.

Parameter	Value range
k_{hop}	$(0.005\text{–}0.04)\ \text{ps}^{-1}$
k_{BD}	$(3.3 \cdot 10^{-5}\text{–}0.04)\ \text{ps}^{-1}$
k_{r}	$0.0002\ \text{ps}^{-1}$
k_{trap}	$(0.0002\text{–}0.02)\ \text{ps}^{-1}$
n_{D}	0.5–15

III. RESULTS

A. Direct problem

Let us first focus on the application of NNs to the direct problem—prediction of the averaged kinetics of the lattice from the model parameter values.

We generated 18 066 samples with different k_{hop}, k_{BD}, k_{trap}, and n_{D} as the input parameters and fitted coefficients of Eqs. (3)–(5) as the output parameters. It is important to note that we ordered the stretched exponents of Eq. (4) in a descending order based on the amplitudes. This was done to reduce the ambiguity of the NN predictions. In addition, we also saved the averaged kinetics for every sample. For each sample, the values of the model parameters were drawn from a uniform distribution with limits listed in Table I. We used the lower and upper bounds of the parameter value ranges (listed in Table I) to normalize the input data using min–max normalization,

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}. \tag{6}$$

We split the data into a training dataset (85%) and a testing dataset (15%). Therefore, the accuracy of the NN predictions was based only on the data that the NN had not seen before. The NN was trained using Adam, adaptive moment estimation optimizer,[24] with a 0.001 learning rate for 100 epochs. The default loss function was defined as a mean squared error (MSE) between the fitted coefficients and the

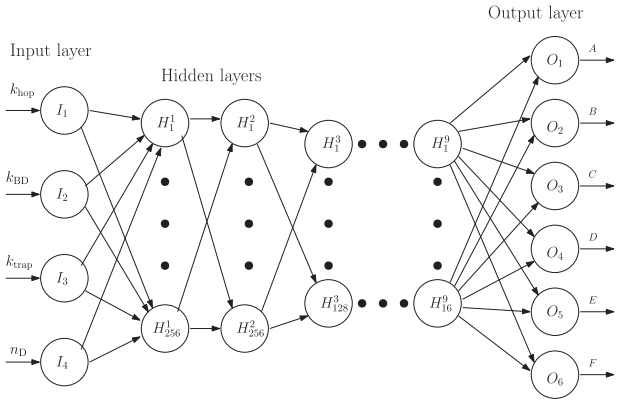


FIG. 2. Schematic illustration of the NN architecture used to predict the excitation dynamics of a 2D lattice system. The lattice model parameters are used as the input, while the output corresponds to the coefficients of the fitting function.

NN predictions. The NNs used in this work were constructed using the *PyTorch* library.[25]

While NNs are a versatile tool, this comes with a cost of having to select a proper network architecture for a particular problem. We tested a variety of different sequential NN architectures, but the best result was achieved with a nine-layer network with ReLu activation functions,

$$\mathrm{ReLu}(x) = \max(0, x), \tag{7}$$

on every layer. First hidden layer had 256 neurons, which were connected to another 256-neuron hidden layer. This was connected with two 128-neuron hidden layers. This pattern was repeated until we reached 16 neurons in the final hidden layer, which was connected with the desired six neurons in the output layer (see Fig. 2).

After testing different fitting functions, we found that the best function for this use is a sum of two stretched exponentials, Eq. (4). The most accurate fit was achieved with Eq. (3), but the NN was not able to accurately predict the coefficients for this function. Using Eq. (4), the obtained fitting coefficient solutions were stable and the average MSE between the fitted function and the averaged kinetics was $1.77 \cdot 10^{-5}$. For illustrations of different MSE values, see Fig. 3. It can be seen that the difference between the curves is barely noticeable when the MSE is of the order of 10^{-5}. When the MSE is of the order of 10^{-4}, there are slight differences between the curves. On the other hand, when the MSE is larger than 10^{-4}, the differences between the curves are substantial.

To evaluate the quality of the NN predictions, we used the MSE between the fitted curve and the NN predicted curve. As mentioned before, at first, the NN training loss function was the MSE between

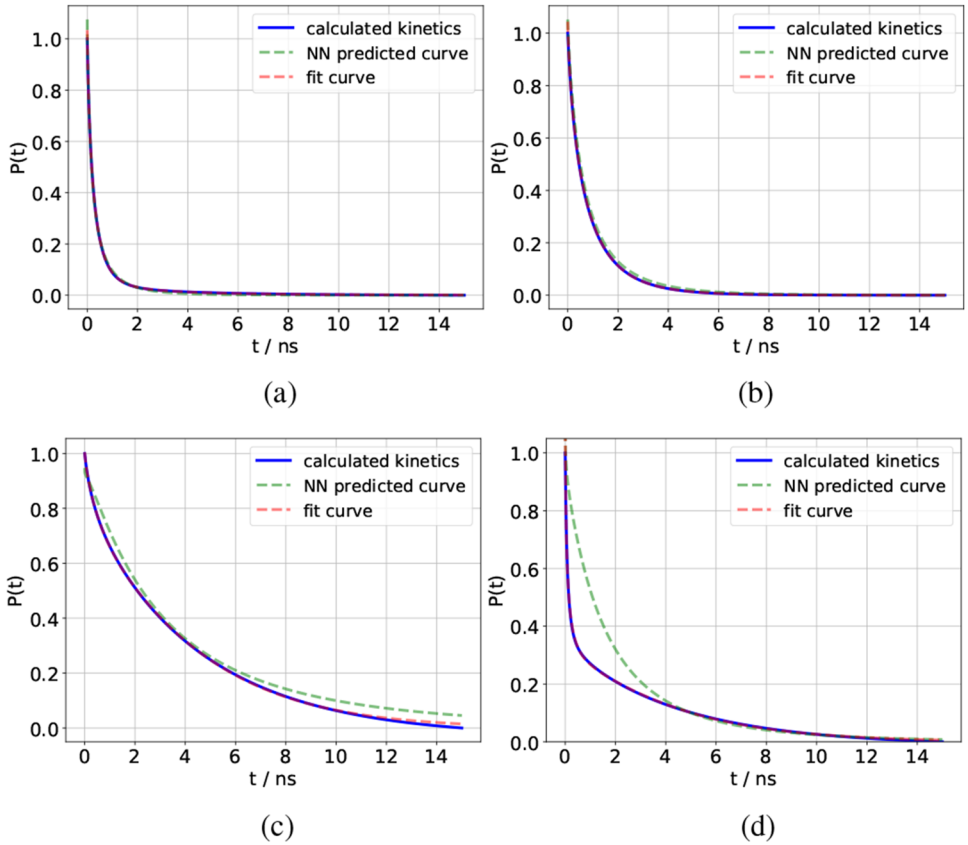


FIG. 3. Illustration of different MSE values between the calculated and fitted or predicted kinetics. (a) Predicted vs. fitted MSE = $1.93 \cdot 10^{-5}$, predicted vs. calculated MSE = $1.98 \cdot 10^{-5}$. (b) Predicted vs. fitted MSE = $1.14 \cdot 10^{-4}$, predicted vs. calculated MSE = $1.16 \cdot 10^{-4}$. (c) Predicted vs. fitted MSE = $1.08 \cdot 10^{-3}$, predicted vs. calculated MSE = $1.12 \cdot 10^{-3}$. (d) Predicted vs. fitted MSE = $1.10 \cdot 10^{-2}$, predicted vs. calculated MSE = $1.11 \cdot 10^{-2}$.

TABLE II. Different loss functions and the MSEs achieved with them. To achieve the accuracy given in the first row, we used the MSE loss function between the fitted coefficients and the coefficients predicted by the NN. The second row was achieved using the L_{con} loss function. The third row is the result of using MSE loss between the averaged kinetics and the predicted kinetics, which were reconstructed from the NN predicted coefficients and the fitting function. The forth row corresponds to the loss function being the sum of two terms used in previous rows. The last used L_{con} with ten times larger dataset.

Loss function	Fitted curve vs NN predicted curve	Averaged kinetics vs NN predicted curve
$\mathrm{MSE}(c_k, p_k)$	$1.85 \cdot 10^{-2}$	$1.85 \cdot 10^{-2}$
L_{con}	$1.40 \cdot 10^{-4}$	$1.53 \cdot 10^{-4}$
$\mathrm{MSE}(P(t), f(p_k))$	$1.68 \cdot 10^{-4}$	$1.93 \cdot 10^{-4}$
$L_{con} + \mathrm{MSE}(P(t), f(p_k))$	$9.77 \cdot 10^{-5}$	$1.21 \cdot 10^{-4}$
L_{con} with 10× data size	$3.09 \cdot 10^{-5}$	$5.94 \cdot 10^{-5}$

the fitted coefficients and the coefficients predicted by the NN. Interestingly, this resulted in clearly inadequate NN predictions, with the MSE between the fitted and the NN predicted curves being very large, $1.85 \cdot 10^{-2}$ (see Table II). This result leads us to investigate other possible loss functions.

Even though the decaying kinetics function can be conveniently parameterized with a small number of parameters, as in Eq. (4), there are often many possible parameter values that can describe the same curve in such description. This could be problematic for the NN to disentangle. Motivated by this, we decided to investigate whether more accurate NN predictions could be achieved by comparing the kinetics obtained from both input and predicted parameters. Thus, we redefined the loss function as a conditional loss,

$$L_{\mathrm{con}} = \sum_{k=1}^{N} \begin{cases} \mathrm{MSE}(p_k, c_k), & \text{if } B_k > 0 \text{ or } E_k > 0, \\ \mathrm{MSE}(f(p_k), f(c_k)), & \text{if } B_k < 0 \text{ and } E_k < 0, \end{cases} \tag{8}$$

where $f(\cdots)$ is the fitting function [Eq. (4)], B_k and E_k are coefficients predicted by the NN, N is the batch size, p_k are the predicted coefficients by the NN, and c_k are the fitted coefficients. We can only calculate loss as the difference of the functions if the functions are not divergent when $t \to \infty$. This is ensured by checking whether predicted coefficients satisfy $B < 0$ and $E < 0$. Using L_{con} as the loss function improved the NN prediction accuracy by an order of magnitude, see Table II. Note that we also tried to force negative B and E values in the NN prediction, but that resulted in worse NN accuracy; thus, such an approach was not adopted.

Encouraged by this success, we have also tried other loss functions. We used the MSE between the averaged kinetics and the NN prediction as the loss function. This resulted in ~5 fold increase in NN prediction quality (see the third line in Table II). In addition,

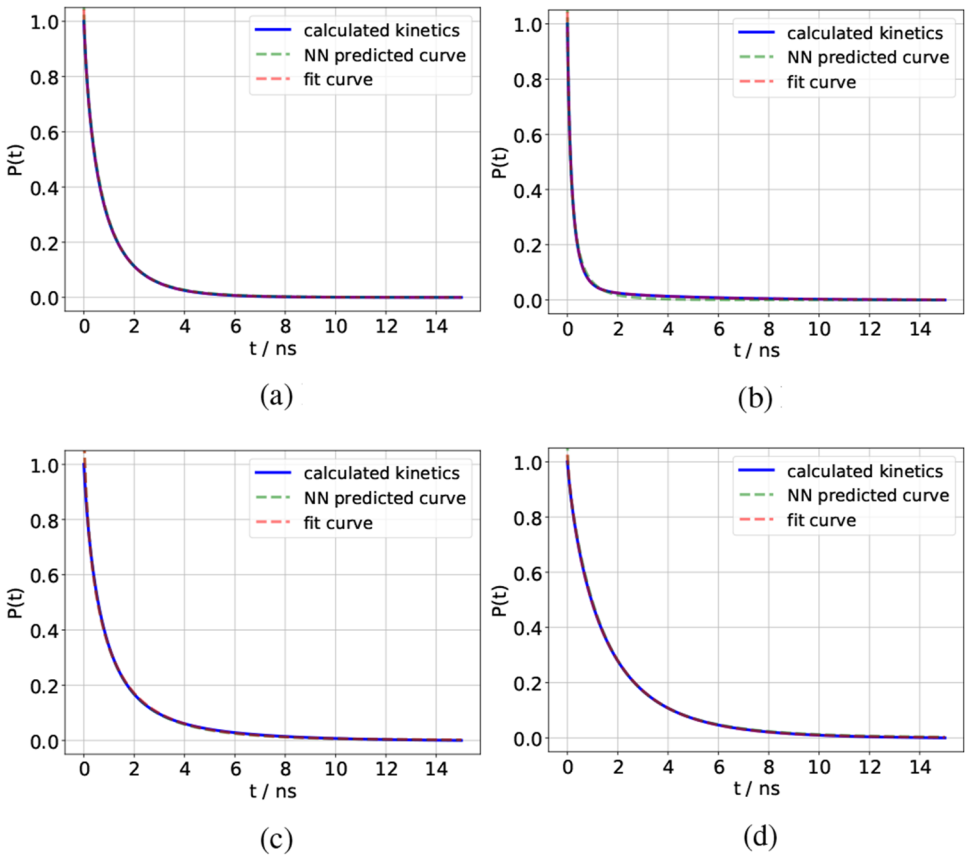


FIG. 4. Examples of averaged kinetics, fitted curves, and NN predicted curves. (a) Predicted vs. fitted MSE = $6.01 \cdot 10^{-6}$, predicted vs. calculated MSE = $8.41 \cdot 10^{-6}$. k_{hop} = $5.68 \cdot 10^{-3}$ ps^{-1}, k_{BD} = $7.36 \cdot 10^{-5}$ ps^{-1}, k_{trap} = $1.34 \cdot 10^{-3}$ ps^{-1}, n_D = 10.47. (b) Predicted vs. fitted MSE = $4.00 \cdot 10^{-5}$, predicted vs. calculated MSE = $4.10 \cdot 10^{-5}$. k_{hop} = $1.56 \cdot 10^{-2}$ ps^{-1}, k_{BD} = $5.67 \cdot 10^{-5}$ ps^{-1}, k_{trap} = $2.08 \cdot 10^{-4}$ ps^{-1}, n_D = 13.17. (c) Predicted vs. fitted MSE = $3.91 \cdot 10^{-6}$, predicted vs. calculated MSE = $5.10 \cdot 10^{-5}$. k_{hop} = $5.59 \cdot 10^{-3}$ ps^{-1}, k_{BD} = $3.72 \cdot 10^{-5}$ ps^{-1}, k_{trap} = $2.50 \cdot 10^{-4}$ ps^{-1}, n_D = 8.47. (d) Predicted vs. fitted MSE = $1.00 \cdot 10^{-5}$, predicted vs. calculated MSE = $1.21 \cdot 10^{-5}$. k_{hop} = $1.00 \cdot 10^{-2}$ ps^{-1}, k_{BD} = $4.50 \cdot 10^{-5}$ ps^{-1}, k_{trap} = $1.82 \cdot 10^{-2}$ ps^{-1}, n_D = 3.05.

we also tried adding this MSE with L_{con}, but this did not result in a noticeable improvement over the latter result (see the fourth line in Table II).

Finally, we tried to train the NN using L_{con} and a ten times larger training dataset, and we achieved MSE ($5.94 \cdot 10^{-5}$) of the same order of magnitude as was obtained when fitting the averaged kinetics ($1.77 \cdot 10^{-5}$); see the last line in Table II. Since, in this case, the training data do not have to include the averaged kinetics, thus saving memory, we hold this choice as our final result. This NN was able to accurately predict the averaged kinetics for all parameter values, as illustrated in Fig. 4.

Next, we investigated how the NN prediction accuracy depends on the size of the dataset. We trained the network with different amount of training data, and the obtained MSE values are presented in Fig. 5. We see that a training dataset with $5 \cdot 10^3$ samples already results in reasonable accuracy. The accuracy improves rapidly upon increasing the size of the dataset to $2 \cdot 10^4$ samples. Further increase in the training dataset size results in somewhat better accuracy. Therefore, we can conclude that it would be optimal to use a dataset of about 40–80 thousands of samples with the same train/test (85%/15%) split. Still, larger training datasets do not result in any additional gain in the NN prediction accuracy. In Fig. 5, we also present the accuracy dependence for the KNeighborsRegressor algorithm, which was the best of all non NN methods. Clearly, it is outperformed by the NN by two to three times, depending on the dataset size. Other tested methods showed even worse accuracy, with the MSE between the predicted curve and averaged kinetics of the order of 10^{-3}. These results justify our focus on NNs in this study.

It is also important to investigate how the accuracy of the NN predictions depends on changes n one single parameter value. To this end, we performed a variation of one of the lattice model parameter value while keeping the other parameter values fixed. The NN was trained on a bigger dataset, which consisted of 101 thousand samples. We have chosen the default value set to be $k_{hop} = 0.011\ ps^{-1}$, $k_{BD} = 0.0004\ ps^{-1}$, $k_{trap} = 0.0011\ ps^{-1}$, and $n_D = 3.82$, and the initial MSE between the NN predicted kinetics and the calculated kinetics was $2.08 \cdot 10^{-5}$. Figure 6 shows the dependence of the accuracy of the NN predictions on the variation of lattice model parameters separately. Since it is of considerable interest whether the NN can extrapolate when encountering parameter values that were not present in the training set, we also checked the parameter values outside the limits listed in Table I. We see that the

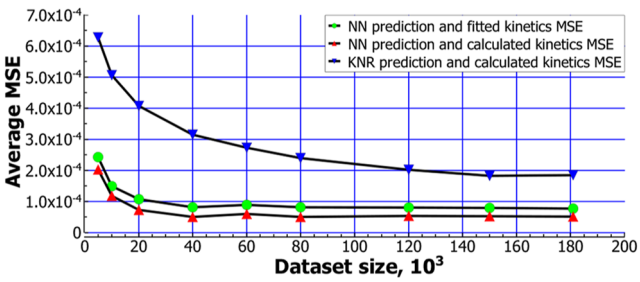
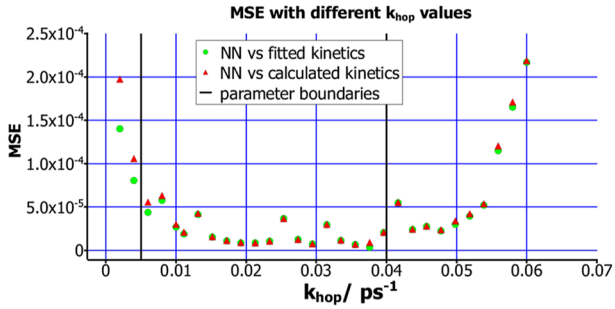
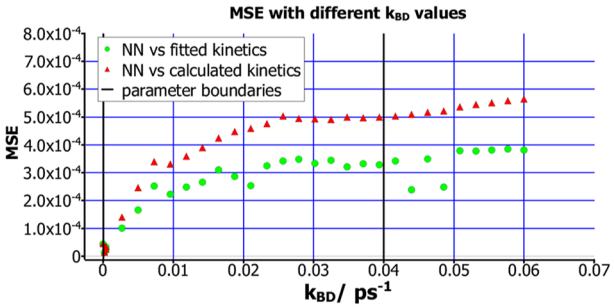


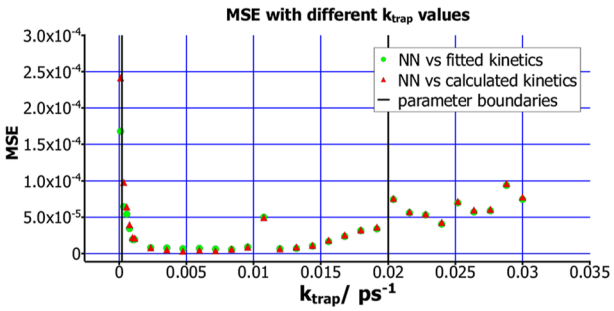
FIG. 5. NN and KNeighborsRegressor prediction accuracy dependence on the training dataset size.



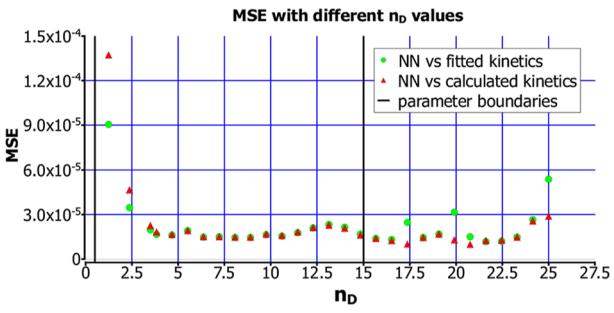
(a)



(b)



(c)



(d)

FIG. 6. NN prediction accuracy dependence on the variation of one lattice model parameter values. Default parameters values set was $k_{hop} = 0.011\ ps^{-1}$, $k_{BD} = 0.0004\ ps^{-1}$, $k_{trap} = 0.0011\ ps^{-1}$, and $n_D = 3.82$, and the initial MSE between the NN predicted kinetics and the calculated kinetics was $2.08 \cdot 10^{-5}$. (a) Accuracy of the NN predictions for different k_{hop} values. (b) Accuracy of the NN predictions for different k_{BD} values. (c) Accuracy of the NN predictions for different k_{trap} values. (d) Accuracy of the NN predictions for different n_D values.

variation of k_{hop} within the boundaries of the training set does not lead to worse accuracy. On the other hand, using values outside the boundaries results in a significantly deteriorating accuracy. The variation of k_{BD} parameter caused the biggest decrease in accuracy. The small decrease in NN prediction accuracy, when we vary the lattice model parameters k_{trap} and n_D, suggests that the NN is able to generalize the predictions for input values in the training data value range and even beyond the upper limit. It is interesting to observe that the NN prediction accuracy is most sensitive to the variation of the k_{hop} parameter, which defines the overall transport timescale of the system.

B. Inverse problem

Now, let us turn to the inverse problem—predicting the values of the lattice model parameters from the averaged kinetics. If compared to the direct problem, the input and the output of the NN are switched. Therefore, we feed the coefficients of the fitting function or the fitting curves themselves to the NN and compare the values of the predicted lattice model parameters with the real lattice model parameter values used to calculate the corresponding kinetics. As mentioned previously, NNs have difficulties interpreting certain fitting functions, and we achieved the most accurate results using Eqs. (4) and (5).

Once again, we first discuss the network architecture. The most straightforward approach would be to reverse the network architecture used for the direct problem. After testing numerous NN architectures, however, we concluded that better accuracy is achieved when a separate NN is created for every lattice model parameter. Thus, four separate NNs taken together are needed to fully characterize the lattice model. The input layer of each NN has the same amount of neurons as there are fitting parameters in the chosen fitting function. As was the case for the direct problem, we used a nine layer network with ReLu activation functions on every layer. The number of neurons in the hidden layers gradually decreases from 256 to 16 and finally reduces to one in the output layer (see Fig. 7). For the optimization of the NNs, we used the Adam optimization algorithm. The NNs were trained for 100 epochs with a 0.001 learning rate.[24] The loss function was defined as the MSE between the predicted and true parameter values. The dataset consisted of 133 thousand samples with 75%/25% train/test split.

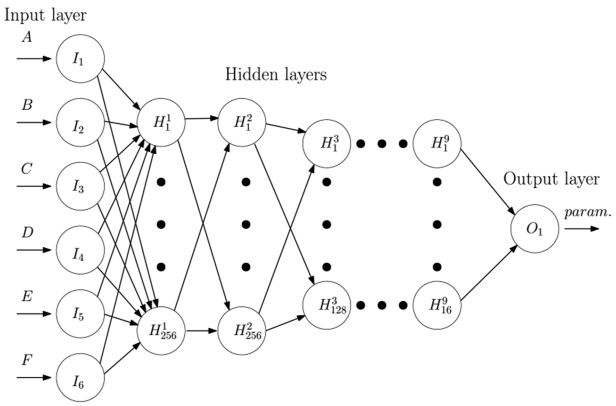


FIG. 7. NN architecture for the inverse problem, if six coefficients from the fitting function of Eq. (4) are used in the input layer. If we use the fitting function of Eq. (5), only the input layer changes to five neurons and the rest of the architecture remains the same.

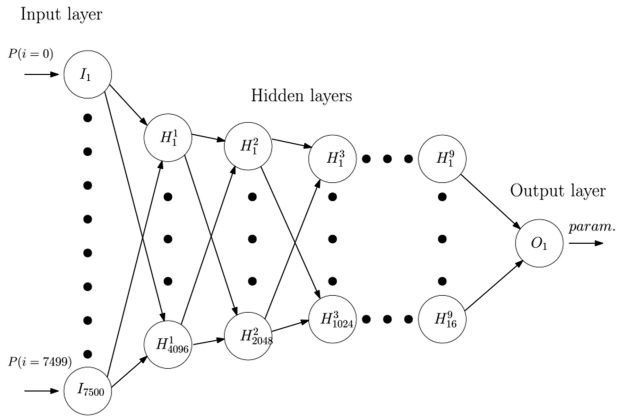


FIG. 8. NN architecture for inverse problem, when using fitting curve as the input. Here, $P(i) = P(i \cdot \Delta t)$ with $\Delta t = 2$ ps.

As another possible strategy, we also tried using the fitted kinetics as the input of the NN. The fitted curve was recreated using the fitting coefficients and consisted of 7500 data points. The number of neurons in the hidden layers decreased from 4096 to 16 and then reduced to 1 in the output layer (see Fig. 8).

Using these NN architectures with Eqs. (4) and (5) fitting functions, we generated predictions and calculated the NN output. To obtain a more intuitively understandable accuracy measure, for each parameter we then calculated the ratio of the MSE to the parameter's value range, listed in Table I, which results in an estimation of relative error. These results are presented in Table III. A number of observations can be made. First, the errors of the NN predictions are not large, on the order of 10%. Second, larger errors are obtained for the parameters that have more influence on the kinetics curve, k_{hop} and n_D. Even for these parameters, however, the ratio of the MSE to the input interval length is less than 14%. Third, the difference between NNs trained with Eqs. (4) and (5) fitting functions is very minor. Fourth, results obtained using the coefficients of the fitted functions are consistently better than those obtained using the fitted curves. This might be related to the fact that, in the latter case, the NN itself has considerably more parameters; thus, a much larger training dataset could be needed to avoid overfitting.

In order to evaluate the accuracy of the set of NNs in solving the inverse problem, we adopted the following procedure. First, we

TABLE III. The ratio of the MSE to the input parameter intervals for the inverse problem.

Input	k_{hop} (%)	k_{BD} (%)	k_{trap} (%)	n_D (%)	Average
5 fit coefficients	13.07	1.96	6.50	10.96	8.12
6 fit coefficients	12.69	2.19	6.92	11.52	8.33
5 coeff. fit curve	13.57	2.99	7.19	12.34	9.02
6 coeff. fit curve	13.49	2.34	7.22	12.94	9.00

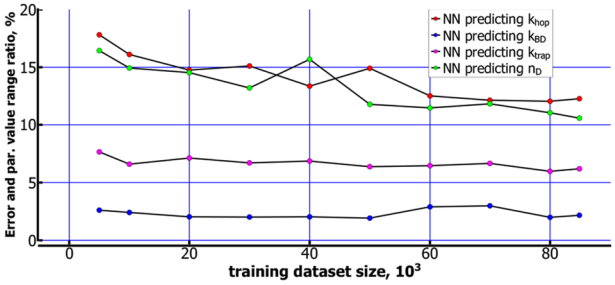


FIG. 9. The set of the most accurate NNs predictions accuracies dependence on the training dataset size for the inverse problem.

predicted 1500 lattice model parameter sets using the most accurate set of NNs (see the first row of Table III). Then, we used the lattice model together with Pauli master equations to calculate the averaged kinetics for every predicted parameter set, which we will denote as $P_{\text{pred}}(t)$. Finally, we calculated the MSE between the true averaged kinetic curve $P(t)$ and $P_{\text{pred}}(t)$, which was $1.31 \cdot 10^{-2}$. The kinetics reconstructed from the predicted lattice model parameters are clearly inaccurate, if we compare them to the initial (true) averaged kinetics. One of the reasons is the limited accuracy of lattice model parameter predictions by the set of sequential NNs. Another reason is that the lattice model, involving the Pauli master equations and averaging through different lattice arrangements, is extremely sensitive to the parameter variations.

Finally, we tried training the set of the most accurate NNs (see the first row in Table III) with different dataset sizes and investigated its effect on the prediction accuracy. These results are presented in Fig. 9. It is interesting to note that contrary to the direct problem, there is no sharp increase in accuracy. Instead, we observe gradual improvement for k_{hop} and n_{D} lattice parameter values that are predicted less accurately. Interestingly, k_{trap} and k_{BD} parameters exhibit no significant change in accuracy upon increasing the dataset size from $5 \cdot 10^3$ to $8.5 \cdot 10^4$ training samples. Overall, it seems that at least $6 \cdot 10^4$ training samples are needed if the best possible accuracy for all parameters is required.

IV. DISCUSSION AND CONCLUSIONS

In this work, we are interested in the applications of artificial NNs to electronic excitation dynamics in 2D lattice systems. In particular, we would like to find out if the NNs are useful for speeding up solutions of a global optimization problem, which has to be solved when we want to fit some specific experimental data using a lattice model. We will first highlight the most important of our results and insights that follow from them. Then, we will move on to discussing the application of the NNs to the aforementioned global optimization problem.

We have successfully trained a deep, fully connected, sequential NN to predict the averaged kinetics of the lattice model from the values of the model parameters. We have achieved excellent accuracy that is similar to that obtained from fitting the modeled kinetics curve with suitable analytical functions. This result is interesting, as it means that the NN has successfully managed to account for the heterogeneity of the lattice system and the required statistical averaging over different random distributions of the dark sites. However, it must be noted that we obtained high accuracy only with a quite deep NN.

We would like to highlight that our investigations demonstrated that a choice of a suitable loss function is crucial. In our case, there is some degeneracy in the input parameters, as functions that we used to fit the calculated kinetics [Eqs. (3)–(5)] are composed of two or three functionally identical terms. This degeneracy makes the training of the NN problematic if the loss function is the MSE of the coefficients of the fitting function. This is evident from a poor agreement between the predicted and fitted kinetics curves. On the other hand, if we change the loss function in training to MSE between the predicted and fitted decay curves, the prediction accuracy of the NN increases substantially. This highlights that the reduced dimensionality data descriptions, while very useful, must be used with care, as usually, it is the actual data, and not its reduced representation that is of interest.

Using the NNs for the inverse problem was less successful than for the direct problem. While the values of the lattice model parameters could be predicted with a reasonable accuracy, the resulting curves differ substantially from the fitted kinetics. This is probably related to the fact that the inverse problem for our lattice model is ill-posed. For example, the limit of infinitely deep traps can be reached when either $k_{\text{trap}} \to \infty$ or $k_{\text{BD}} \to 0$. Thus, for large values of k_{trap}, the value of k_{BD} does not affect the kinetics, while the value of k_{trap} ceases to matter when k_{BD} is very small.

Let us now turn our attention to the application of the NNs to the solution of the global optimization problem. There are two possible strategies for applying NNs in the case considered here. One would be to use the NN trained for the direct problem as a replacement for full simulation using the lattice model. This change should speed up the calculation of the objective function during the solution of the global optimization problem. The other choice would be to use the NN trained for the inverse problem to obtain the lattice model parameter values, bypassing the global optimization problem altogether.

First, let us estimate the numerical effort required using the straightforward approach without NNs. In order to solve the global optimization problem accurately, a population-based method is usually the best choice. If we assume the DE method, the size of the population is usually ten times the number of parameters; thus, for the present case of four parameters, we would have a population of size 40. Assuming 500 population generations that should ensure the convergence to the global minimum, $2 \cdot 10^4$ of calculations of the averaged kinetics $P(t)$ are required.

Applications of NNs involve a few steps. First, the training data have to be generated. Second, the NNs have to be trained. Third, the best possible NN has to be selected for the problem at hand. Often, it is the first step that is the most expensive computationally. Therefore, we will concentrate on the numerical cost of that step. We have demonstrated that for direct prediction of excitation kinetics, a dataset of at least $4 \cdot 10^4$ samples should be used for training. Our estimations imply that if only a single experimental dataset needs to be fitted based on such a lattice model, applications of NNs would not help to reduce the required calculation time. Nonetheless, it is often the case that a single model is checked for the applicability to several similar experimental datasets collected under different conditions. For example, in Ref. 15, we applied a similar lattice model

for the description of time-resolved fluorescence spectra of LHCII aggregates at nine temperatures. Continuing our estimations, this would correspond to $1.8 \cdot 10^5$ calculations required, exceeding by far the number of samples needed to train the NN. Therefore, we can reasonably conclude that in such a case applications of NN should be beneficial.

Considering the inverse problem, our results show that at least $6 \cdot 10^4$ training samples should be used for the best possible accuracy. Unfortunately, the NN predicted model parameter values do not result in a kinetics curve that is close to the input data. Thus, it may appear that the NN trained for the inverse problem is not very useful. That is not the case because we could exploit the fact that the predicted values of the model parameters are quite close to the actual ones. Thus, a population-based global optimization algorithm, like DE, could be set up with significantly narrower bounds for parameters, which should result in a much faster and, therefore, computationally less expensive convergence.

Finally, let us discuss a possible strategy for the application of the NNs to solving the global optimization problem using the lattice model. Let us assume that several experimental datasets have to be fitted with the same model. The key point to recognize is that using the population-based optimization algorithms, like DE, generates a lot of data, which usually goes to waste. If instead we save that data, it could be used as a training dataset for an NN. Thus, we suggest that the first experimental dataset should be fitted with a population-based algorithm with reasonably wide bounds for parameters, and the calculation data should be saved. Then, while the second experimental dataset is being fitted, a suitable NN could be constructed. Since the first dataset should already be fitted, the NN could be easily tested with respect to accuracy and speed-up of calculations. If the NN is trained successfully, the rest of the experimental datasets could be fitted using the NN, thus saving computational time and resources.

Clearly, the strategy presented is based on a somewhat simplified picture of the NN application, and some possible issues (regarding, e.g., suitable network architecture) might arise, complicating the process. Nonetheless, our experience shows that even straightforward application of population-based methods also often involves several starts and stops with different bounds for parameters or a number of iterations. Therefore, applications of NN should be beneficial in a real-world scenario.

Let us stress that here we focused on the application of NNs to population kinetics. In many quantum mechanical systems that are probed by ultrafast spectroscopy methods, such as two-dimensional optical spectroscopy,[26–28] effects of quantum coherence are also of importance. The dynamics of such open quantum systems are described by much more complicated master equations than Eq. (1). Thus, it is interesting to note recent work where NNs were used for the inverse problem of recovering the system parameters from its calculated dynamics.[29] Therefore, we expect a growing interest in applications of NNs for such problems in the foreseeable future.

ACKNOWLEDGMENTS

This work was supported by the Research Council of Lithuania (LMTLT Grant No. S-MIP-20-44). Computations were performed using the resources of the supercomputer "VU HPC" at the Faculty of Physics, Vilnius University.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Pranas Juknevičius: Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Visualization (equal); Writing – original draft (equal). **Jevgenij Chmeliov**: Conceptualization (equal); Supervision (equal); Writing – review & editing (equal). **Leonas Valkunas**: Conceptualization (equal); Funding acquisition (equal); Writing – review & editing (equal). **Andrius Gelzinis**: Conceptualization (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

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

REFERENCES

[1] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).
[2] P. O. Dral, J. Phys. Chem. Lett. **11**, 2336 (2020).
[3] J. Zhang, Y.-K. Lei, Z. Zhang, J. Chang, M. Li, X. Han, L. Yang, Y. I. Yang, and Y. Q. Gao, J. Phys. Chem. A **124**, 6745 (2020).
[4] C. Lu, Q. Liu, Q. Sun, C.-Y. Hsieh, S. Zhang, L. Shi, and C.-K. Lee, J. Phys. Chem. C **124**, 7048 (2020).
[5] A. Farahvash, C.-K. Lee, Q. Sun, L. Shi, and A. P. Willard, J. Chem. Phys. **153**, 074111 (2020).
[6] F. Häse, C. Kreisbeck, and A. Aspuru-Guzik, Chem. Sci. **8**, 8419 (2017).
[7] S. Bandyopadhyay, Z. Huang, K. Sun, and Y. Zhao, Chem. Phys. **515**, 272 (2018).
[8] L. E. Herrera Rodríguez and A. A. Kananenka, J. Phys. Chem. Lett. **12**, 2476 (2021).
[9] M. Rodríguez and T. Kramer, Chem. Phys. **520**, 52 (2019).
[10] M. S. Chen, T. J. Zuehlsdorff, T. Morawietz, C. M. Isborn, and T. E. Markland, J. Phys. Chem. Lett. **11**, 7559 (2020).
[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).
[12] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms* (Springer, 2008).
[13] M. Clerc, *Particle Swarm Optimization* (John Wiley and Sons, 2010).
[14] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution. A Practical Approach to Global Optimization*, Natural Computing Series (Springer-Verlag, Berlin, 2005).
[15] J. Chmeliov, A. Gelzinis, E. Songaila, R. Augulis, C. D. P. Duffy, A. V. Ruban, and L. Valkunas, Nat. Plants **2**, 16045 (2016).
[16] A. Gelzinis, J. Chmeliov, A. V. Ruban, and L. Valkunas, Photosynth. Res. **135**, 275 (2018).
[17] V. Mascoli, A. Gelzinis, J. Chmeliov, L. Valkunas, and R. Croce, Chem. Sci. **11**, 5697 (2020).
[18] N. Tessler, Y. Preezant, N. Rappaport, and Y. Roichman, Adv. Mater. **21**, 2741 (2009).

[19]A. A. Kurilovich, V. N. Mantsevich, Y. Mardoukhi, K. J. Stevenson, A. V. Chechkin, and V. V. Palyulin, Phys. Chem. Chem. Phys. **24**, 13941 (2022).
[20]L. G. Boulu, L. K. Patterson, J. P. Chauvet, and J. J. Kozak, J. Chem. Phys. **86**, 503 (1987).
[21]G. Yang, N. Wu, T. Chen, K. Sun, and Y. Zhao, J. Phys. Chem. C **116**, 3747 (2012).
[22]S. Caffarri, K. Broess, R. Croce, and H. van Amerongen, Biophys. J. **100**, 2094 (2011).
[23]R. E. Blankenship, *Molecular Mechanisms of Photosynthesis*, 2nd ed. (Wiley Blackwell, Chichester, 2014).
[24]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[25]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, in *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035.
[26]A. Gelzinis, R. Augulis, V. Butkus, B. Robert, and L. Valkunas, Biochim. Biophys. Acta **1860**, 271 (2019).
[27]E. Collini, J. Phys. Chem. C **125**, 13096 (2021).
[28]S. Biswas, J. Kim, X. Zhang, and G. D. Scholes, Chem. Rev. **122**, 4257 (2022).
[29]K. Naicker, I. Sinayskiy, and F. Petruccione, Phys. Rev. Res. **4**, 033175 (2022).