Observing how deep neural networks understand physics through the energy spectrum of one-dimensional quantum mechanics

Kenzo Ogure
Department of Nuclear Engineering, Kyoto University,
Kyoto daigaku-katsura, Nishikyo-ku, Kyoto, 615-8540, Japan

We investigated how neural networks(NNs) understand physics using one-dimensional quantum mechanics. After training an NN to accurately predict energy eigenvalues from potentials, we used it to confirm the NN's understanding of physics from four different aspects. The trained NN could predict energy eigenvalues of a different potential than the one learned, focus on minima and maxima of a potential, predict the probability distribution of the existence of particles not used during training, and reproduce untrained physical phenomena. These results show that NNs can learn the laws of physics from only a limited set of data, predict the results of experiments under conditions different from those used for training, and predict physical quantities of types not provided during training. Since NNs understand physics through a different path than humans take, and by complementing the human way of understanding, they will be a powerful tool for advancing physics.

I. INTRODUCTION

In recent times, deep neural networks (DNNs) have made remarkable progress in image recognition, natural language processing, voice recognition, and anomaly detection through numerous technological breakthroughs. Among these breakthroughs, the residual connection prevents gradient loss, even when neural networks (NNs) have deep layers, contributing to the very high image recognition capability[1]. Another example is the attention mechanism, which has succeeded in connecting neurons in distant locations, a shortcoming of convolutional neural networks (CNNs), and has significantly advanced fields such as translation, where relationships between distant words are essential[2]. In addition to accuracy improvement, NNs have begun generating new images or sentences by themselves, and their performance has been rapidly improving [3, 4]. In a slightly different field, the combination of DNNs and reinforcement learning has rendered humans incapable of competing with computers in table games such as Go and Shogi, where human intuition was previously superior[5, 6].

The significant difference between NNs and previous artificial intelligence is that NNs do not require human intuition at the design stage. Earlier types of artificial intelligence relied on humans to predetermine features of objects and perform learning. In contrast, NNs do not require human intuition. In return, NNs comprise numerous parameters and require high computing power for learning. In addition, it is difficult for humans to understand how a trained NN operates. Another advantage of NNs is that the input and output can be exchanged in some cases, enabling their application to inverse problems.

The properties of not requiring prior intuition and applicability to inverse problems make NNs promising in natural sciences, where the goal is to reveal unknown problems, and NNs have been used in many fields, including physics[7]. In this paper, the subject of this study is the Schrödinger equation in quantum mechanics, and even if we focus only on the surrounding area, we can see many recent developments related to NNs. Partial differential equations, including the Schrödinger equation, have been solved using NNs[8], potentials have been estimated inversely from wave functions[9, 10], and soliton solutions to the nonlinear Schrödinger equation have been investigated using NNs[11]. A new family of NNs inspired by the Schrödinger equation has also been proposed[12].

In these studies, the Schrödinger equation was used as known, and in this sense, human understanding of physics was already assumed. However, since we consider this study preparation for applying NNs to unsolved physical systems, we do not directly solve the Schrödinger equation using NNs. From this viewpoint, the research conducted in a context most similar to this paper is Ref.[13]. This pioneering study provided NNs with only two-dimensional (2D) potentials and the corresponding energy eigenvalues of the ground states. This prior work and our study differ in technical aspects, such as the method for generating the potentials and NN's architecture, but the most significant difference is that we are most interested in how NNs understand physics. In this sense, despite the differences in physical systems and methods, our study shares a common interest with Ref.[14].

The application of NNs to concrete physics problems seems to be progressing well. However, how an NN learns physics is embedded in numerous parameters it contains and is difficult to find. In particular, whether the NN is merely learning its output pattern or understanding the underlying physics is essential in considering the future use of NNs. It should also be clarified what the NN is focusing on to understand physics and whether we can extract different kinds of information from the NN than the fed data. This study sheds light on these points using one-dimensional (1D) quantum mechanics as a subject. Since our research assumes that we will use NNs to solve unknown physics

problems in the future, we only provide NN potentials and a few energy eigenvalues, which are physical quantities that can be measured experimentally.

This paper is organized as follows. First, in Sec.II, after a brief introduction to NNs for physicists, we demonstrate how to generate datasets and training results in this paper. Then, in Sec.III, where we state our main results, we investigate four aspects of how NNs understand physics and the potential for NNs to be used in physics research. Next, section IV summarizes the results of this paper and provides some insights into the future potential of NNs in physics. Finally, the NN architecture used in this paper is summarized in the appendix.

II. SETUP AND TRAINING OF NEURAL NETWORKS

This section first presents a brief introduction to NNs for physicists and then explains how NNs are trained in this paper on 1D quantum mechanics problems in Sec.II A. Next, we explain how to prepare the datasets in Sec.II B, and finally, we present the training results in Sec.II C.

A. Application of Neural Networks to one-dimensional Quantum Mechanics

Regression using an NN is a kind of variational method if we use a term familiar to physicists. An NN is a function that contains parameters, which are determined to reproduce the best ground-truth outputs. This function, inspired by the brain's structure, has several characteristics:

- While the variational trial function in physics is typically assumed to be an elementary function or a superposition of elementary functions by some physical intuition, NNs do not require such intuition. In return, NNs have far more parameters than trial functions in traditional variational methods in physics to be sufficiently flexible.
- Since NNs are developed by a composite function of linear transformations and simple functions called activation functions, the back-propagation technique can quickly obtain their derivatives.
- As a technical matter, NNs are beginning to be firmly recognized for their usefulness, and supportive libraries such as Pytorch and Tensorflow have been developed, as well as technologies such as Cuda for faster computation on GPUs. This has enabled experimenting with NN technology to solve different problems.

Here, we explain the training of NNs in more detail. Training an NN requires a dataset recorded through experience, experimentation, or artificial data generation. We write this as $\{(v_m^{(k)}, E_i^{(k)})\}$, where $\{v_m^{(k)}\}$ is real-valued vectors of dimension M, and $\{E_i^{(k)}\}$ is a real-valued vector of dimension i_{max}. The index k is the sample's label, which can take values from 1 to N. The dataset size N should be as large as possible to train NNs.

An NN can be written as $f_i(\{v_m^{(k)}\};\{w_\alpha\})$. In addition to taking $\{v_m^{(k)}\}$ as an argument, it has real numbers $\{w_\alpha\}$ as parameters. The majority of $\{w_\alpha\}$ is the weights and bias of the linear combination used to create the composite function, but other parameters can also be included. Training an NN means optimizing $\{w_\alpha\}$ so that $f_i(\{v_m^{(k)}\};\{w_\alpha\})$ is as close to $E_i^{(k)}$ as possible for most of $\{v_m^{(k)}\}$.

The flow of this optimization is depicted in FIG.1. First, when the input data $\{v_m^{(k)}\}$ is fed to the NN, the output $f_i(\{v_m^{(k)}\};\{w_\alpha\})$ is obtained. Then, the loss function resulting from the error between the output of this NN and the ground-truth dataset is calculated. In addition, the first-order derivative to the parameter $\{w_\alpha\}$ can be calculated almost simultaneously using a method called back-propagation. The value of $\{w_\alpha\}$ is updated according to the loss function derivative to $\{w_\alpha\}$. Repeating this process until the loss function does not significantly decrease is called NN training.

In actual training, instead of adding up all samples in the loss function, a few samples are selected, which is called batch learning. The batch size can be decided, but it is often determined around the order of 100. Batch learning saves computer memory, and it is empirically known that batch learning is faster than taking all data. The final results are often better when the batch size is smaller. At least, we do not know the general theory for determining the optimal batch size and have empirically determined a good value by trial and error.

Next, we explain how we apply NN learning to a physical problem in this paper. We deal with quantum mechanics in one dimension. Particles are confined to the interval $-1 < x < 1$ by infinitely high walls. Then, we generate N various potentials between $-1 < x < 1$ and take their values at equally spaced points as input data $\{v_m^{(k)}\}$. For the output data $\{E_i^{(k)}\}$, we take 10 of the smallest energy eigenvalues corresponding to the potentials ($i_{max} = 10$). The following subsection explains how to generate the potentials and calculate the corresponding eigenvalues.

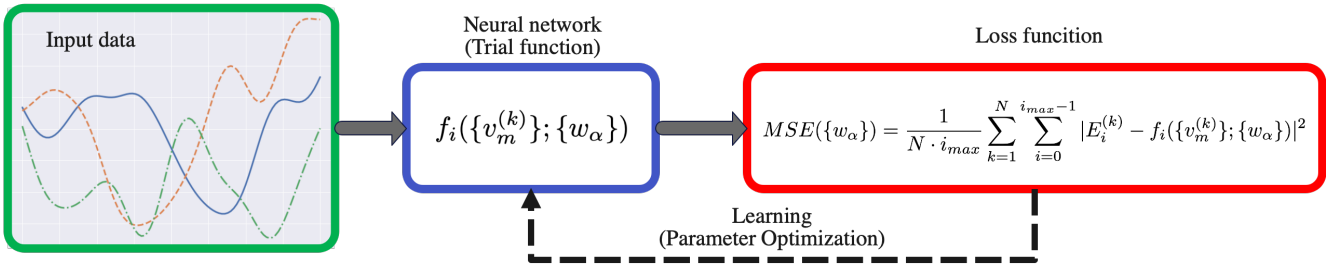

FIG. 1. Training an NN. An NN is a function, which takes a multidimensional real-valued input and produces a multidimensional real-valued output. Training an NN requires many input and ground-truth output datasets. NN learning is the optimization of parameters called weights and biases, $\{w_\alpha\}$, to reduce the loss function, $MSE(\{w_\alpha\})$, linked to the difference between the correct data and the prediction by the output of the NN. In this optimization, the back-propagation method, which can quickly calculate the derivative of the loss function to parameters, is indispensable. In this paper, the input is 1D potentials' values on mesh points, and the output is the corresponding energy spectra. We describe the architecture of our NN and the specific method for optimizing its parameters in the appendix.

We use a 1D CNN. In addition, we also use the residual connection and the self-attention layer, which are effective when used in conjunction with CNNs in the natural language processing and image generation fields[1, 15]. We adopt the mean squared error(MSE) between the ground-truth energy eigenvalues and the outputs from the NN for the loss function. We describe the architecture of our NN and the specific method for optimizing its parameters in the appendix.

B. How to prepare the datasets

We take the energy spectrum of a particle confined to an interval in one dimension as our output data. Hence, we consider infinite potential wells with deformed bottoms as follows,

$$V^{(k)}(x) = \begin{cases} v^{(k)}(x) & (|x| \leq 1) \\ \infty & (|x| > 1) \end{cases} \tag{1}$$

where $v^{(k)}(x)$ are arbitrary continuous functions and k takes values from 1 to N, which is the size of datasets.

Next, we use the kernel method to generate the potential functions, $v^{(k)}(x)$. We use the kernel method only to generate the potential functions, not to perform regression using the Gaussian process. We use the NN for regression. We set mesh points on $-1 < x < 1$ with $2/M$ width as $x_l = (2l - M)/M$ $(l = 0, 1, 2, \cdots M)$ to use the kernel method. Let the kernel function be $k(x, x')$, and then, under our discretization, the kernel matrix is the $(M + 1) \times (M + 1)$ matrix defined by $K_{lm} = k(x_l, x_m)$. By using this kernel matrix as the covariance matrix, we can probabilistically generate the potential function, as follows:

$$\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{2}$$

where $\mathcal{N}$ represents the normal distribution function, with the mean being $\mathbf{0}$ and the covariance matrix being $\mathbf{K}$. Furthermore, $\boldsymbol{v}$ is an $(M + 1)$-dimensional vector, whose components are regarded as potential values at $\{x_l\}$. We obtain $v^{(k)}(x_l)$ by performing this potential generation for N times.

This study uses four kernel functions to generate the training data: Gaussian kernel $k^{(RBF)}(x, x')$, Matérn5 kernel $k^{(M5)}(x, x')$, Matérn3 kernel $k^{(M3)}(x, x')$, and exponential kernel $k^{(Exp)}(x, x')$. These functions are defined, as follows:

$$k^{(RBF)}(x, x') = \sigma^2 \exp\left(-\frac{r^2}{2L^2}\right) \tag{3}$$

$$k^{(M5)}(x, x') = \sigma^2 \left(1 + \frac{\sqrt{5}r}{L} + \frac{5r^2}{3L^2}\right) \exp\left(-\frac{\sqrt{5}r}{L}\right) \tag{4}$$

$$k^{(M3)}(x, x') = \sigma^2 \left(1 + \frac{\sqrt{3}r}{L}\right) \exp\left(-\frac{\sqrt{3}r}{L}\right) \tag{5}$$

$$k^{(Exp)}(x, x') = \sigma^2 \exp\left(-\frac{r}{L}\right) \tag{6}$$

FIG. 2. Potentials generated using the kernel method. The kernel parameters were taken to be the correlation distance $L = 0.2$ and the amplitude $\sigma = 100$ for all kernels. The smoothness varies depending on the kernel used. (a) Potentials generated from the Gaussian kernel, Eq.(3) or Eq.(7) at $\nu = \infty$. These are infinite-order differentiable. (b) Potentials generated from the Matérn5 kernel, Eq.(4) or Eq.(7) at $\nu = \frac{5}{2}$. These are second-order differentiable. (c) Potentials generated from the Matérn3 kernel, Eq.(5) or Eq.(7) at $\nu = \frac{3}{2}$. These are first-order differentiable. (d) Potentials generated from the exponential kernels, Eq.(6) or Eq.(7) at $\nu = \frac{1}{2}$. These are not differentiable.

where we defined $r = |x - x^{'}|$. These functions have two parameters: σ, which controls the magnitude of the generated potential, and L, which controls the correlation distance of the potential. These kernel functions can be written concisely using the gamma function $\Gamma(\nu)$, and the Bessel function of the second kind K_ν, as follows:

$$k_\nu(x, x^{'}) = \frac{2^{1-\nu}\sigma^2}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{L} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{L} \right) \tag{7}$$

The Gaussian, Matérn5, Matérn3, and the exponential kernels correspond to the cases of $\nu = \infty$, $\frac{5}{2}$, $\frac{3}{2}$, $\frac{1}{2}$, respectively. Kernels with larger ν generate smoother potential functions, and the generated potentials are $[\nu]$ times differentiable. Here $[\nu]$ is the largest integer not greater than ν. We show the potentials generated from each kernel in FIG.2, and the smoothness of the potentials changes as ν increases.

Now that we have generated the potentials, which are the input data needed to train the NN, we need to find the energy spectrum of those potentials, the output data. For this purpose, we solve the time-independent 1D Schrödinger equation,

$$\left\{ -\frac{1}{2}\frac{d^2}{dx^2} + V^{(k)}(x) \right\} \psi^{(k)}(x) = E^{(k)} \psi^{(k)}(x), \tag{8}$$

where $\psi^{(k)}(x)$ is the wave function and we use Planck's constant $h = 2\pi$, and the particle's mass $m = 1$. Since we are considering a deformed infinite well potential Eq.(1), the boundary conditions for the wave function are $\psi^{(k)}(\pm 1) = 0$. In the case of the square potential, $v^{(k)}(x) = 0$, the energy eigenvalues are,

$$E_i^{(k)} = \frac{\pi^2(i+1)^2}{8} \quad (i = 0, 1, 2, \cdots), \tag{9}$$

and the corresponding wave functions are,

$$\psi_i^{(k)}(x) = \sin\left[\frac{\pi}{2}(i+1)(x+1)\right]. \tag{10}$$

Since we use the ten lowest energy eigenvalues as the ground-truth output data, it should have a rich structure. To achieve that, we take $\sigma = 100$ in Eqs.(3)-(6) throughout this paper, referring to Eq.(9). Figure 3 shows that this setting works in the actual numerical experiment results.

We use the matrix method[16] to solve this eigenvalue problem for general $v^{(k)}(x)$. This method is appropriate for our setting because it enables us to find multiple energy eigenvalues and corresponding wave functions simultaneously. Furthermore, since we have already discretized the problem to generate the values of the potentials on the mesh points, we can directly use this method.

Applying the matrix method to Eq.(8) is straightforward: rewriting Eq.(8) by replacing the second-order derivative with a difference results in the following matrix eigenvalue problem,

$$H^{(k)}\Psi^{(k)} = E^{(k)}\Psi^{(k)} \tag{11}$$

which we defined, as follows, ($\Delta x = 2/M$, $v_m^{(k)} = v^{(k)}(x_m)$),

$$H^{(k)} = -\frac{1}{2(\Delta x)^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & -2 & 1 \\ 0 & 0 & 0 & \cdots & 1 & -2 \end{pmatrix} + \begin{pmatrix} v_1^{(k)} & 0 & 0 & \cdots & 0 & 0 \\ 0 & v_2^{(k)} & 0 & \cdots & 0 & 0 \\ 0 & 0 & v_3^{(k)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \ddots & v_{M-2}^{(k)} & 0 \\ 0 & 0 & 0 & \cdots & 0 & v_{M-1}^{(k)} \end{pmatrix} \tag{12}$$

Here, we have used the boundary conditions $\psi^{(k)}(x_0) = \psi^{(k)}(x_M) = 0$, so the matrix $H^{(k)}$ is an $(M-1) \times (M-1)$ dimensional matrix and the eigenvector $\Psi^{(k)}$ is $(M-1)$-dimensional vector. This eigenvalue problem is easy to solve because $H^{(k)}$ is a tridiagonal real symmetric matrix. We denote the eigenvectors and corresponding eigenvalues as $E_i^{(k)}$ and

$$\Psi_i^{(k)} = \begin{pmatrix} \psi_i^{(k)}(x_1) \\ \psi_i^{(k)}(x_2) \\ \psi_i^{(k)}(x_3) \\ \vdots \\ \psi_i^{(k)}(x_{M-2}) \\ \psi_i^{(k)}(x_{M-1}) \end{pmatrix}, \qquad \Psi_i^{(k)T}\Psi_i^{(k)} = 1 \tag{13}$$

, respectively. Note that this normalization of the wave function is natural as a normalization for the matrix eigenvalue problem but deviates by $\sqrt{\Delta x}$ from the ordinary normalization condition of the wave function, $\int_{-1}^{1} |\Psi^{(k)}(x)|^2 dx$. There are M-1 eigenvalues, labeling them as $E_i^{(k)} < E_j^{(k)}$ for $i < j$. Since this is 1D quantum mechanics, the energy eigenvalues have no degeneracy. We adopt ten eigenvalues($i = 0, 1, 2 \cdots, 9$) for the ground-truth output data of the NN.

We have depicted one potential generated from the Gaussian kernel and the ten lowest energy eigenvalues of the potential in FIG.3(c). We note that we have confirmed that these are the ten lowest energy eigenvalues by counting the nodes of the corresponding wave functions.

This subsection explains how to create a set of $(M+1)$-dimensional potential vectors, $\{v_m^{(k)}\}$, as input data and 10-dimensional energy eigenvalues, $\{E_i^{(k)}\}$, as output data for training an NN. Here we used the Schrödinger equation to generate these data. We also obtained the eigenvectors, but we do not feed them to the NN. We only feed the NN with the set of potentials $\{v_m^{(k)}\}$ and energy eigenvalues $\{E_i^{(k)}\}$ as if we can obtain only them in a physical experiment.

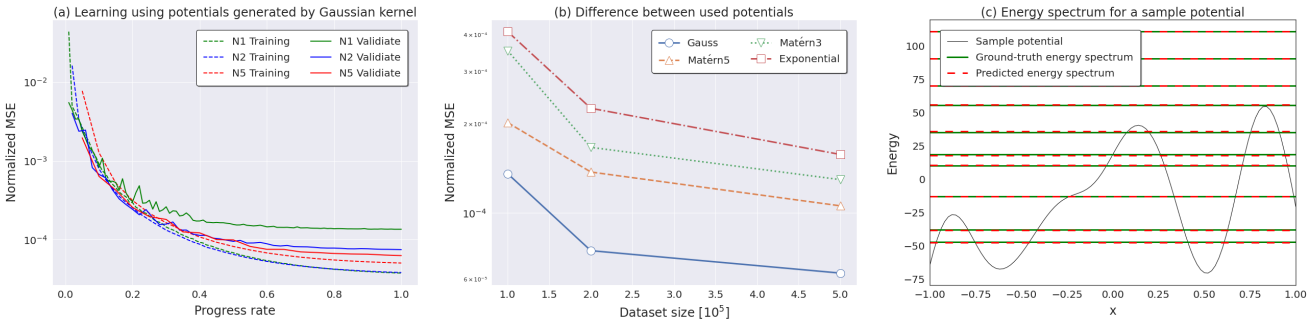

FIG. 3. Training NNs. (a) NMSE in Eq.(14) is plotted against the training progress. The dataset used for training varies between 1×10^5(N1), 2×10^5(N2), and 5×10^5(N5) to study the difference in learning progress. The total number of parameter updates is 1×10^5 in all cases. The dashed line represents the error for the training data, and the solid line represents the error for the validation data. For validation, we used data generated from the same kernel that generated the data used for training. This figure shows that the larger the dataset, the less overfitting there is. (b) The final epoch errors are displayed for each kernel that generates data. This figure shows that smooth data are easier to learn. (c) For one sample potential generated from the Gaussian kernel, we compared the energy eigenvalues predicted by the NN with the ground-truth values calculated from the matrix diagonalization in Eq.(11). As shown in (b), for the data generated from the Gaussian kernel, the average error of the energy eigenvalues is less than 1%, which can barely be seen in this figure.

C. Learning setup and results

We generated training datasets of sizes 1×10^5(N1), 2×10^5(N2), and 5×10^5(N5) for all four kernels of Eq.(3)-(6) to train the NN. The reason why we constructed datasets of different sizes was to examine the degree of overfitting. In addition, we generated different datasets of sizes 1×10^4 for all kernels as validation data. The parameters of the kernels were $\sigma = 100$ and $L = 0.2$. Since the batch size was fixed at 100, one epoch of training would update the NN parameters $\{w_\alpha\}$, 1×10^3, 3×10^3, and 5×10^3 times for dataset sizes N1, N2, and N5, respectively. To keep the total number of parameter updates constant and compare the learning status fairly, we trained 100, 50, and 20 epochs for data N1, N2, and N5, respectively. This setup resulted in the parameter $\{w_\alpha\}$ being updated 1×10^5 times in total for any dataset. The loss function is the MSE. The design of the NN and the precise optimization method of the parameter $\{w_\alpha\}$ are included in the appendix.

Figure 3(a) shows the learning progress. Here, we defined the normalized mean squared error(NMSE) to evaluate the error as a dimensionless quantity, as follows,

$$NMSE(\{w_\alpha\}) = \frac{\sum_{k=1}^{N} \sum_{i=0}^{i_{max}-1} (E_i^{(k)} - f_i(\{v_m^{(k)}\}; \{w_\alpha\}))^2}{\sum_{k=1}^{N} \sum_{i=0}^{i_{max}-1} (E_i^{(k)})^2}. \tag{14}$$

In addition, we defined the progress rate as the ratio of the performed update times of $\{w_\alpha\}$ to the total one, 1×10^5. This figure displays the NN's learning process using datasets generated from the Gaussian kernel. The dashed line represents the error for the training data, and the solid line represents the error for the validation data. This figure shows that overfitting is suppressed as the size of the dataset increases, and the effect of overfitting is minimal at N5. When the size of the dataset is N5, the NMSE is less than 10^{-4} at the end of the training, and the error in the energy eigenvalue is less than 1% on average. We have confirmed the datasets generated from other kernels show the same trend even though some quantitative differences exist.

Figure 3(b) shows the difference in the NMSE after the completion of training depending on the kernel that generates the training data. Validation data were generated from the same kernel that generated the training data. This figure shows that the error is minor for the dataset with a larger size. In addition, the smoother the potential, the smaller the error.

Figure 3(c) shows the prediction of the energy eigenvalues of one potential generated from the Gaussian kernel by an NN trained on data generated from the same kernel with the dataset size N5. This figure shows that the deviation of the prediction of the energy eigenvalue is small.

III. UNDERSTANDING PHYSICS WITH NEURAL NETWORKS

In Sec.II, we showed that NNs can now predict energy eigenvalues accurately, but it is still unclear whether they understand the physics described by Eq.(8). As described in the Appendix, our NN has 4.9×10^6 real number parameters, $\{w_\alpha\}$, and the real number size of the output of the largest dataset, $\{E_i^{(k)}\}$, is 5×10^6. This fact may lead us to suspect that the NN does not understand physics but memorizes the output.

In this section, which describes the main results of this paper, we will show in four aspects that NNs do understand physics and that they are helpful for the study of physics. First, in Sec.III A, we show that NNs can predict energy eigenvalues even for potentials of a different shape than the training data, and in Sec.III B, we show that NNs focus on the same region of the potential as humans do. Next, in Sec.III C, we successfully extract information about the eigenvectors not used to train the NN from the trained NN. Finally, we show that NNs can predict unknown physical phenomena in Sec.III D.

These results show that NNs can learn the laws of physics from only a limited set of data, predict the results of experiments under conditions different from those used for training, and predict physical quantities of types not provided during training. Since NNs understand physics through a different path than humans take, and by complementing the human way of understanding, they will be a powerful tool for advancing physics.

A. Expand the scope of applicable data

In Sec.II C, the training performance of the NN was verified with validation data generated from the same kernel that generated the training data. However, if the NN understands the physics governed by Eq.(8), it should predict the energy eigenvalues of a potential generated from a different kernel than the one that generated the training data. Therefore, this subsection shows how the NN predicts energy eigenvalues for the validation data generated from a different kernel than the training data generated.

Figure 4(a) shows the training progress using the dataset generated from the Gaussian kernel. Throughout this section, the dataset size is fixed at N5. This figure shows that the error for the validation data generated from the Gaussian kernel is the smallest, but the prediction is not broken, even for datasets generated from other kernels.

We show the error at the final epoch in FIG.4(b) by the kernel that generated the training data. This figure shows that NNs trained with data such as the one in FIG.2(d), generated from the exponential kernel, show promising results, regardless of the dataset type. On the other hand, the NN trained on the dataset generated from the Gaussian kernel is not good at predicting potentials that are not smooth. To express this in a motto-like way, it indicates that if the NN solves complex problems in practice, it can solve easy problems in the actual test, but if it only solves easy problems in practice, it cannot solve complex problems in the actual test.

We used the NN trained on the dataset generated from the exponential kernel in FIG.4(c) to predict the energy eigenvalues of the same potential as in FIG.3(c), generated from the Gaussian kernel. This figure shows that, although there is a slightly significant deviation in the energy eigenvalue of the seventh excited state, the overall prediction is accurate. This is because the NN has seen only the jagged potential in FIG. 2(d), understood the physical laws governing this system, and calculated the energy eigenvalues of the smooth potential in FIG. 2(a).

The findings in this subsection show that NNs can help physics experiments produce more results with fewer practical resources. For example, even when only materials with rough surfaces are available, it would be possible to train an NN with the results of experiments using those materials and then use the trained NN to predict the results of experiments using materials with smooth surfaces. In addition, for an experiment with an ample parameter space, we can experiment with several parameter regions and use the observed results to train the NN. Then, we may raise candidates for parameter regions, where interesting physical phenomena would be observed, by using the trained NN to search for unknown parameter regions.

B. The Intuition of Neural Networks

What would a physicist consider when guessing the energy eigenvalues of the potential in FIG.3(c)? Most physicists would first approximate the vicinity of the two minima using quadratic functions and, from the curvatures, predict some energy eigenvalues in the harmonic oscillator approximation. Next, they may focus on two maxima and consider that energy eigenvalues sufficiently larger than these would approach those represented by Eq.(9). Does an NN have this kind of intuition based on experience? In this subsection, we focus on the attention map in an NN and observe the intuition that the NN has.

The attention mechanism was introduced to overcome the weaknesses of CNNs and has frequently been used recently. A convolutional layer scans small convolutional kernels over the language or image to extract information,

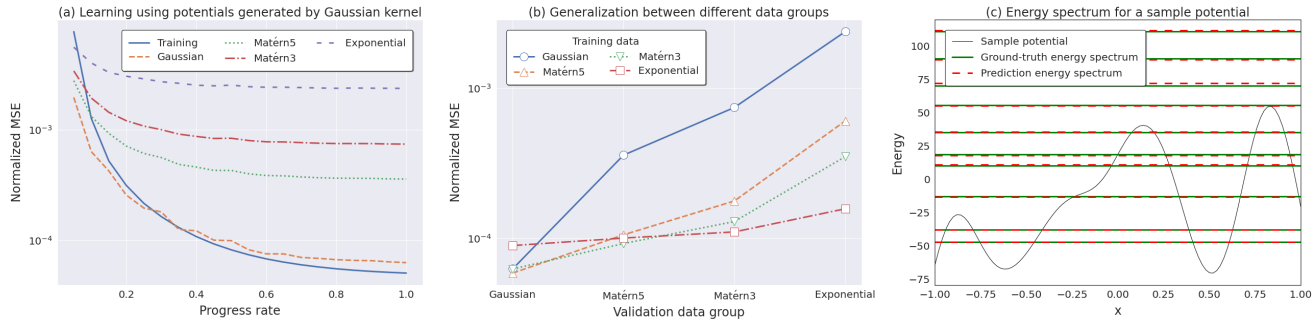

FIG. 4. Validation using a dataset generated from different kernels from the training data generated. (a) NMSE versus learning progress when using data generated from the Gaussian kernel for training data. The predictions for data generated from other kernels are also good, and it seems that predictions for smooth potentials are easier than rough ones. (b) The error at the final epoch is shown for each data used for training. The size of the dataset is 5×10^5. The NN trained with the exponential kernel shows a good prediction for any validation dataset. (c) The NN trained with the exponential kernel was used to predict energy eigenvalues for the same potential as in FIG.3(c). The energy eigenvalues of the seventh excited state are slightly different, but considering that the prediction is for a completely different potential from the training one, the prediction is excellent.

which enables efficient learning with few parameters. Among methods familiar to physicists, wavelet analysis is the closest to CNNs, but the significant difference is that the convolutional kernel changes its shape as the NN learns. The weak point of this method is that it is impossible to connect information about points far apart in position. This weakness has been fatal in the field of natural language processing, where the relationship between distant words is critical.

The attention mechanism can compensate for this weakness. It learns the information considered closely related to the folded information as an attention map and can relate information located far from each other. This advantage has dramatically improved the accuracy of translation in natural language processing[2, 17]. Moreover, this attention mechanism has also been used in image generation, achieving good results[15]. We have incorporated this attention mechanism into our NN, and, in this subsection, we focus on the obtained attention maps. The appendix shows how self-attention layers are embedded in the NN.

Figure 5 shows the first and second self-attention maps for the potential in FIG.3(c). We used an NN trained with N5 data generated from the exponential kernel. To make the figure easier to understand, we aligned the potential shapes on the left and top sides of the self-attention maps. This self-attention map represents the importance of the relationship between two points on the potential learned by the NN. The figure on the left is the self-attention map placed in the layer closest to the input data and shows that the NN perceived the relationship between two minima of the potential as essential. The figure on the right is a self-attention map placed in the second self-attention layer. The NN perceived the relationship of maxima locations as meaningful in addition to the relationship of minima locations. Intriguingly, this perception is the same as physicists' intuition, as mentioned at the beginning of this subsection.

We checked potentials not shown here, and it also seems that NNs consider the minima of potentials the most important factor, followed by the maxima. The self-attention maps at higher levels did not show as clear a trend as the first two self-attention maps, probably due to the effects of repeated convolution and pooling layers.

It turns out that NNs look in the same places as humans to understand the laws of physics from limited data. It is fascinating that the kind of intuition we observed in NNs matches that of humans. We know that the Schrödinger equation, Eq.(8), governs the problem we are dealing with in this paper, but we do not have a unified understanding or a beautiful approximate model for many physical systems. Even in such a case, it is possible to train an NN using experimental data, and if we can obtain information about what the trained NN considers essential, it could be an excellent hint for finding a unified law or creating a beautiful approximate model for that physical system.

C. Elicit information that is not being fed to the neural network

When we trained the NN, we only gave it information about the potential and the corresponding energy eigenvalues. However, if the NN understands the physics described by Eq.(8) in this process, it may extract information other than the energy eigenvalues. This subsection attempts to find the probability of the particle's existence, which is not used in training, from a trained NN.

The function $f_0(\{v_m\};\{w_\alpha\})$ of the learned NN predicts the energy eigenvalues of the ground state using values at each point of the potential. If $f_0(\{v_m\};\{w_\alpha\})$ sensitively changes when the value of the potential at a point, v_m,

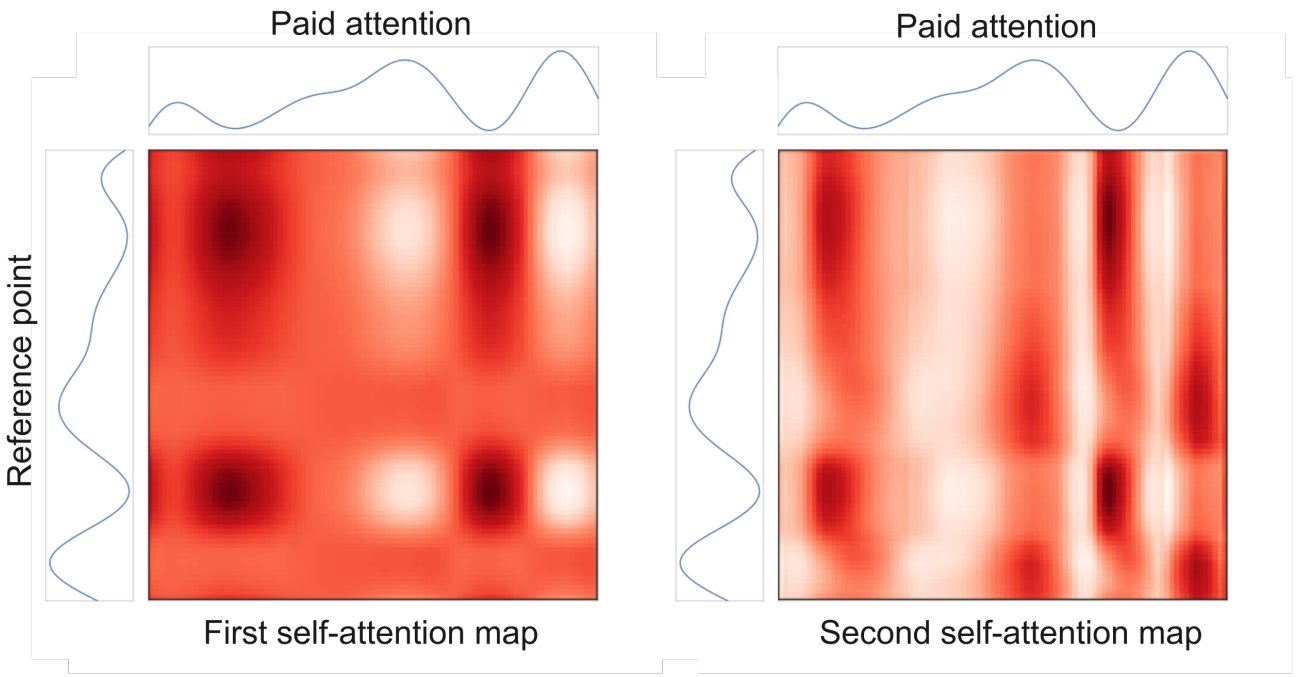

FIG. 5. Attention maps for the potential in FIG.3(c). The figure on the left shows the attention map of the first self-attention layer in the NN. The top and left sides of the figure show the corresponding potential shape. We can see which potential points the NN considers relevant for each reference point on the left side. The NN is intensely interested in the relationship between the minima of the potentials. The figure on the right shows the attention map of the NN's second self-attention layer. In this attention map, we can see that the NN is interested in the relationship between minima and maxima.

is changed, there is a high probability that there is a particle in the vicinity. Therefore, the first-order derivative of $f_0(\{v_m\};\{w_\alpha\})$ to the potential value is considered the existence probability of the particle for the ground state. The same can be said for excited states.

This can also be explained, as follows: $f_i(\{v_m\};\{w_\alpha\})$ should have been learned to approximate E_i's representation. From the discretized quantum mechanics, Eq.(11)-(13), the first-order derivative of the energy eigenvalue to the potential value is the square of the eigenvector, as shown below,

$$\frac{dE_i}{dv_m} = \frac{d}{dv_m}\left(\Psi_i^T H \Psi_i\right) = \Psi_i^T \left(\frac{d}{dv_m} H\right) \Psi_i + E_i \frac{d}{dv_m}\left(\Psi_i^T \Psi_i\right) = \psi_i^2(x_m). \tag{15}$$

This should result in the following relationship if the learning is sufficient:

$$\left.\frac{\partial f_i(\{v_m\};\{w_\alpha\})}{\partial v_m}\right|_{\{w_\alpha\}=\text{learned values}} \simeq \psi_i^2(x_m). \tag{16}$$

As a property of NNs, the first-order derivative of the argument can be easily obtained using back-propagation. When training an NN, only the derivative to the parameter $\{w_\alpha\}$ is used, not the derivative to the input data $\{v_m\}$, but, here, we reverse the position and fix the parameter $\{w_\alpha\}$ on the trained values and obtain the derivative to the input data $\{v_m\}$.

The left and right sides of Eq.(16) are displayed in FIG.6. The NN was trained with N5 data generated from the exponential kernel. The derivative values were not normalized again, and the values were displayed as they were.

Although there are some unnatural parts, such as negative values, the general agreement regarding the location of the peaks and the number of nodes is excellent. This result of obtaining the probability distribution of the particle's existence is surprising, considering that the used NN is only given the jagged potential such as FIG.2(d) and the energy eigenvalues.

This subsection shows that the trained NN provides information other than the output data used during training. As a technical aspect, we also found that differentiation on input data using back-propagation, which is not typically used, may be helpful in physics.

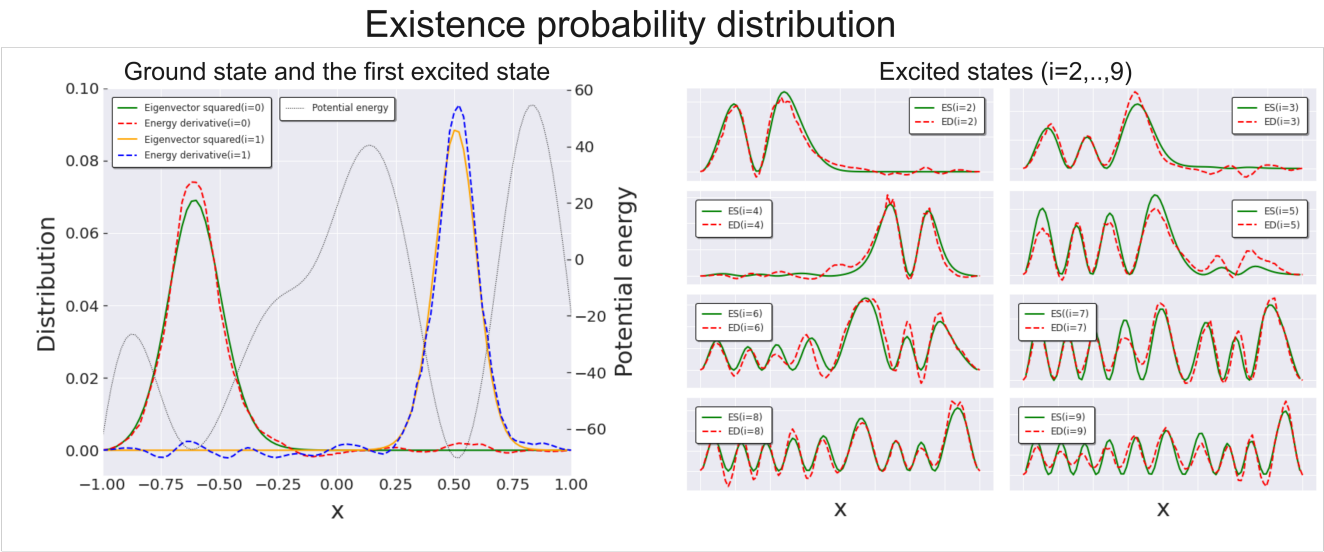

FIG. 6. Probability distribution of particle existence estimated from a neural network. Both sides of Eq.(16) are displayed on the left for the ground state and the first excited state. The same quantities are displayed in the figure on the right for an even higher excited state.

D. Predicting Physical Phenomena with Neural Networks

This last subsection examines whether NNs can predict physical phenomena from limited experimental data. Here, we only reproduce well-known physical phenomena, but in principle, the same procedure should be able to predict phenomena in various physical systems for which there are no unified laws or good approximation models.

1D quantum mechanics has no degeneracy of energy eigenvalues, as can be seen from the fact that even the symmetric double-well potential has no degeneracy of energy eigenvalues. Two energy eigenvalues do not intersect, and their eigenstates cross over when the potential changes slowly and continuously. This phenomenon plays an essential role in the MSW effect of the solar neutrino problem[18–21].

The following setup is used to reproduce this phenomenon. First, we generate two potentials, $v^{(1)}(x)$ and $v^{(2)}(x)$, using the Gaussian kernel, and subtract the energy eigenvalues of the respective ground states to create potentials $\tilde{v}^{(1)}(x)$ and $\tilde{v}^{(2)}(x)$ adjusted so that the energy eigenvalues of the ground states become zero. We then introduce a real mixing parameter $\lambda \in [0, 1]$ and create a potential $v(x)$ that varies continuously from the linear combination, as follows:

$$v(x) = (1 - \lambda)\tilde{v}^{(1)}(x) + \lambda\tilde{v}^{(2)}(x). \tag{17}$$

This continuously varying potential is shown in FIG.7(a).

Figure 7(b) shows the predicted energy eigenvalues from the NN as a function of λ. The energy eigenvalues are changing without crossing. The NN trained with N5 data generated from the exponential kernel was used.

The energy eigenvalues of the ground state and the energy eigenvalue of the first excited state are comparable around $\lambda = 0.8$. We investigate what is happening in the vicinity of this value using the method of Sec.III C. Figure 8(a) shows the existence probability distribution of the particle at $\lambda = 0.75$. The ground state is on the left and the first excited state is on the right. Figure 8(b) shows the existence probability distribution of the particle at $\lambda = 0.79$. The existence probability distributions of the ground state and the first excited state overlap. Figure 8(c) shows the existence probability distribution of the particle at $\lambda = 0.83$. The ground state is on the right, and the first excited state is on the left. In this way, we can see how the eigenstates continuously cross over.

This subsection showed that NNs could understand physical laws from jagged potentials and their energy eigenvalue data and reproduce physical phenomena on smooth potentials. In this way, NNs can predict physical phenomena before experiments are conducted.

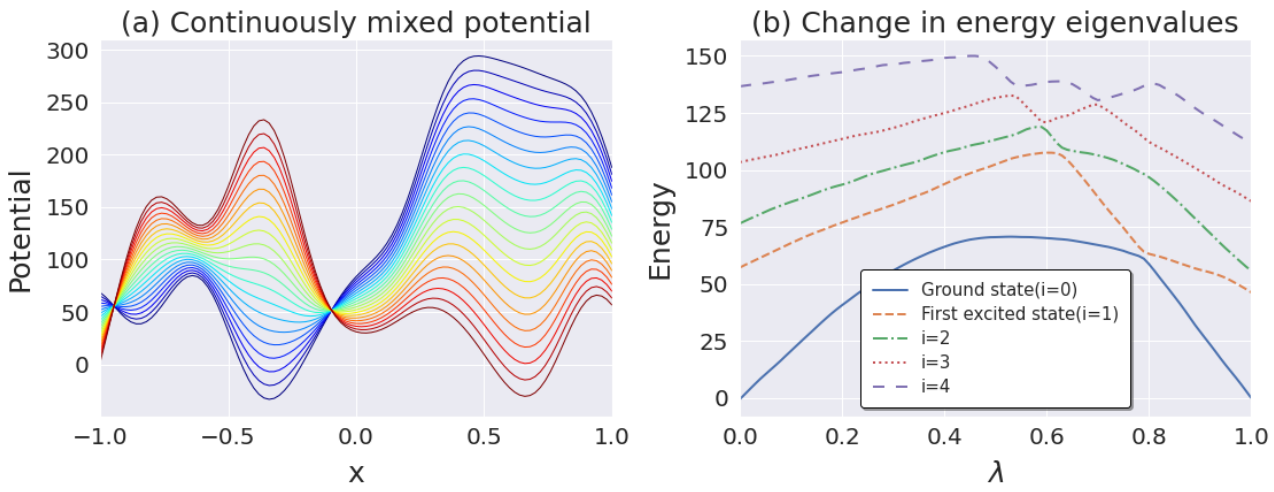

FIG. 7. Continuously varying potential and corresponding energy eigenvalue changes. (a) Two Gaussian kernel-generated potentials adjusted to zero ground state energy were linearly combined with a mixing constant λ. (b) An NN trained with the data generated from the exponential kernel was used to predict the evolution of the energy eigenvalues of the potentials generated in (a). The energy eigenvalues are changing without degenerating.

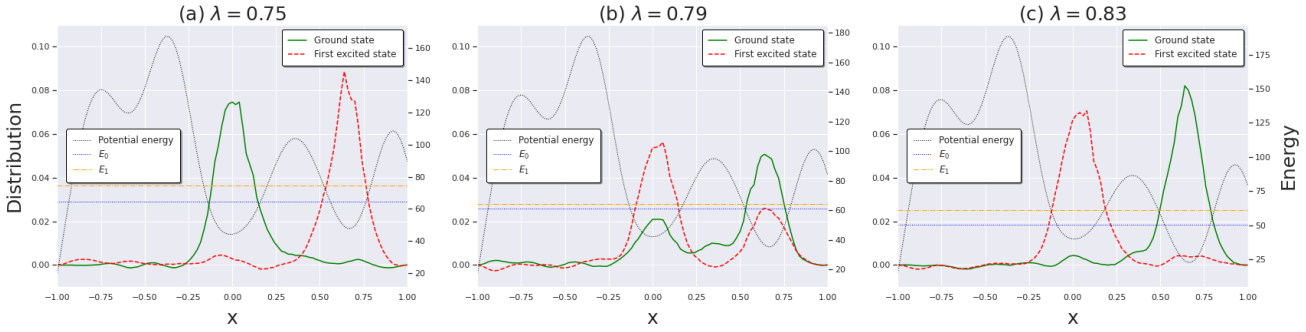

FIG. 8. Crossing over of states. (a) The probability of the existence of a particle at $\lambda = 0.75$. (b) Existence probability of the particle at $\lambda = 0.79$.(c) Existence probability of the particle at $\lambda = 0.83$. We used the left side of Eq.(16) for this estimation.

IV. CONCLUSION

This paper investigated how NNs understand physics using 1D quantum mechanics as a subject. In the preparation stage in Sec.II, it was found that a DNN can find energy eigenvalues from only potentials with errors of approximately 1%.

We showed from four aspects that NNs could understand physics and are helpful for physics research in Sec.III.

- The NN could predict energy eigenvalues, even for potentials different from the one used for training.
- The NN focused on the exact locations of potentials that humans see when they attempt to predict energy eigenvalues at a glance
- The NN could predict the existence probability distribution of particles not used during training.
- The NN predicted the crossover of states when the potential changes slowly and continuously.

As shown in FIG.9, how NNs and humans understand physics seems different. Humans understand physics using their intuition from experimental data, discover unifying laws such as the Schrödinger equation that governs physical systems, or build phenomenological approximation models. On the other hand, NNs attempt to understand physical systems by adjusting numerous parameters, taking advantage of their high computational power to explain a large amount of experimental data. Although humans and NNs follow different paths to understand physics, the underlying physics is the same, and, as a result, they can make the same predictions.

The understanding of physical systems by NNs has the following advantages.

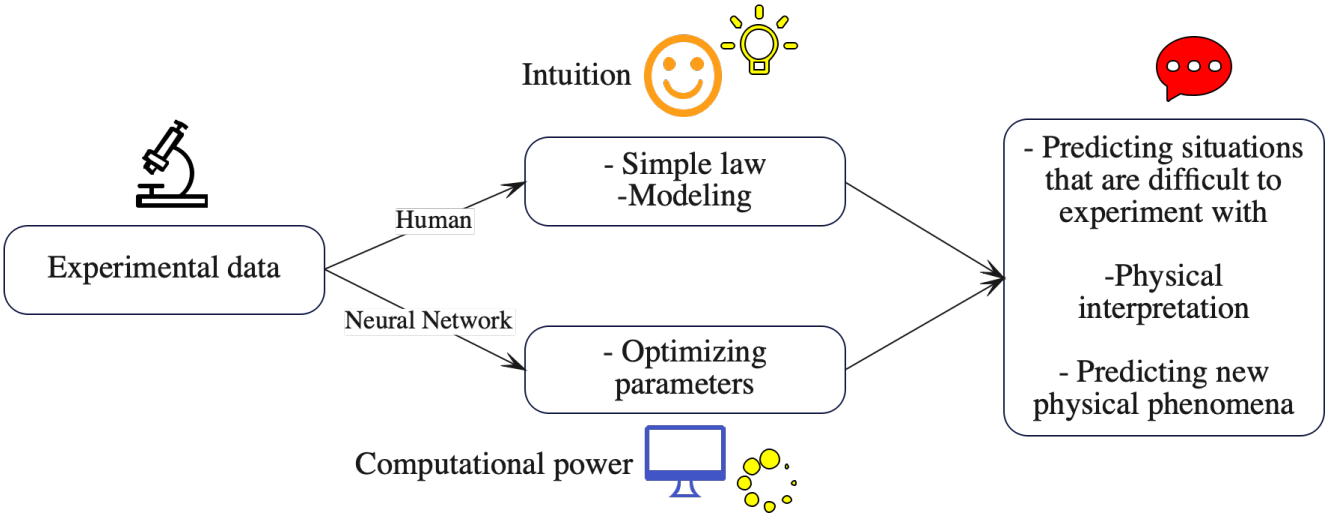

FIG. 9. Differences in the understanding of physics by humans and neural networks. Humans look at the results of physical experiments and use their intuition to derive concise physical laws and create models that describe physical systems well. On the other hand, neural networks use significant computational power to optimize numerous parameters from the results of physical experiments. Both can make predictions about physics, but the process is entirely different.

- It allows us to tackle problems without prior intuition. The NN used in this paper is a combination of layers used in natural language processing and image recognition, and no part of it has been specially devised for physics.
- It is possible to predict the results of experiments under different conditions than those used for training without creating a unified law or a reasonable approximation model.

On the other hand, it has the following drawbacks.

- Massive experimental data are required to create an NN with high reproducibility while preventing overfitting.
- It is possible to improve the accuracy of the approximation, but it does not seem very easy to obtain the exact solution so far.

These results show that NNs can learn the laws of physics from only a limited set of data, predict the results of experiments under conditions different from those used for training, and predict physical quantities of types not provided during training. Since NNs understand physics through a different path than humans take, and by complementing the human way of understanding, they will be a powerful tool for advancing physics.

Appendix: Architecture of Neural Networks

This appendix describes the architecture of the NN we used in this paper. Our NN consists of a combination of the Resnet and SAGAN structures, which have recently been used with success in natural language processing and image generation[1, 15]. The residual connection used in Resnet mitigates gradient vanishing and increases the depth of the NN. On the other hand, the self-attention mechanism used in SAGAN mitigates the shortcoming that convolutional layers cannot relate to distant neurons and is very powerful in natural language processing and image generation.

We began with the overall structure, as shown in FIG.10. First, the input is a 101-dimensional vector corresponding to the potential value at each point. Next, this input is stretched in the channel direction by the blocks containing the convolutional layer, reducing it to 17 dimensions in the spatial direction, whereas it becomes 256 dimensions in the channel direction. Finally, it is flattened to a 1D vector and passed through full connection layers, outputting a 10-dimensional vector. This 10-dimensional vector corresponds to 10 small energy eigenvalues.

Figure 11 shows the details of the Res-SA block used in FIG.10. The input first passes through the convolutional layer and then passes through the residual block three times. Finally, a self-attention layer is used to relate the information about distant locations. We embedded the self-attention layer in the same way as in the SAGAN study[15]. The only difference is that SAGAN needed to flatten a 2D image to 1D to create the attention map, but our 1D potential can skip this process.

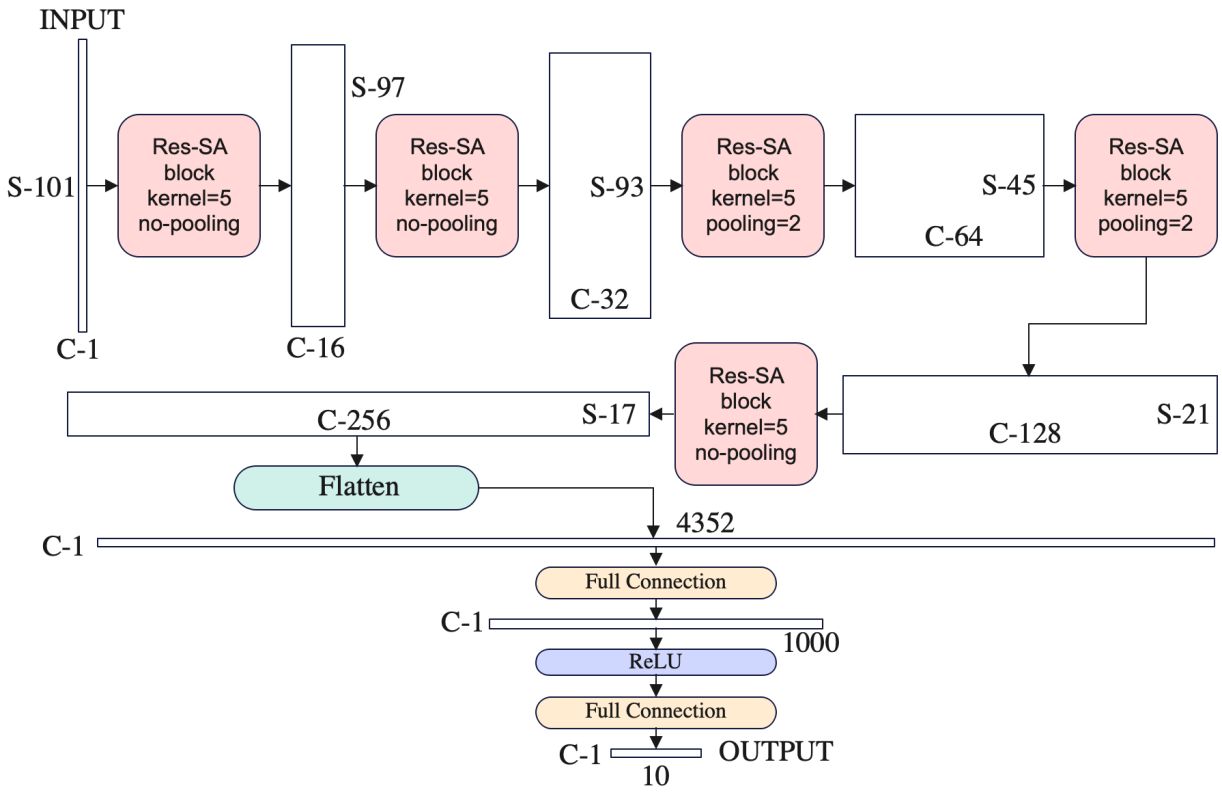

FIG. 10. Design of the entire neural network. The rectangles show the shapes of the tensor at that position. First, the input is the potential value at 101 points in a one-dimensional space. Then, passing through the Res-SA block five times, the tensor becomes longer in the channel direction and shorter in the spatial direction. Finally, it is transformed into a flattened 1-dimensional tensor, which goes through full connection layers and becomes a 10-dimensional vector. This 10-dimensional vector corresponds to the values of the ten lowest energy eigenvalues. The Res-SA block in this neural network is depicted in detail in FIG.11.

In Fig12, we show the structure of the residual block. This block consists of three convolutional layers and one residual connection and has the feature that the input and output have the same shape. This structure is the same as the one used in Resnet.

We have employed the ReLU function for all activation functions, totaling 31 times. The total number of parameters used in this network is approximately 4.9×10^6.

Next, we explain the specific method of learning. We employed Adam as the optimization method and varied the learning rate from 1×10^{-3} (initial value) to 1×10^{-5} (value at the last epoch) in a geometrical progression. In addition, since learning is more efficient when the input and output values are of order O(1), we multiplied the input and output values by 0.01 and adjusted them to be of that magnitude. The batch size was fixed to 100. Pytorch was the NN framework used, and the GPy library provided the kernel functions used. The Numpy library generated the random numbers from the covariance matrix. For diagonalization of the matrix, we used the Scikit-learn library.

[1] K. He, X. Zhang, S. Ren, and J. Sun, in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016) pp. 770–778.
[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, in *Advances in neural information processing systems* (2017) pp. 5998–6008.
[3] D. P. Kingma and M. Welling, arXiv:1312.6114 (2013).
[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Advances in neural information processing systems **27** (2014).
[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, Nature **529**, 484 (2016).

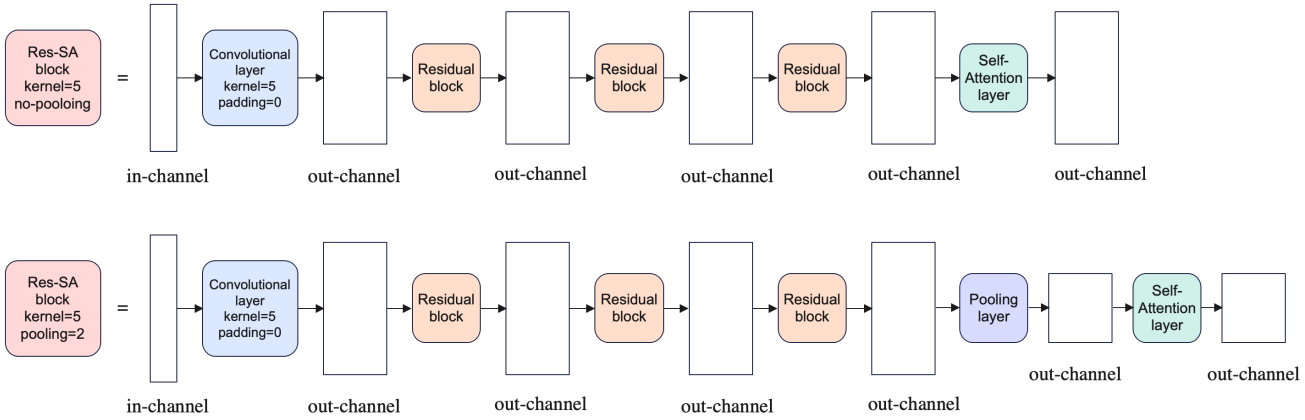

FIG. 11. Res-SA block design. The input tensor can change the number of channels in the initial convolution layer. At that time, the kernel size is 5, and the padding size is 0, so the tensor size in the spatial direction is reduced by 4. If there is a pooling layer, the tensor size in the spatial direction is halved there. For the other layers, the size of the tensor does not change. The deep structure increases the flexibility of the function, and the self-attention layer helps relate the information of points far apart in position. For the self-attention layer, we used the same structure as in the SAGAN paper[15]. The details of the residual block are depicted in FIG.12.

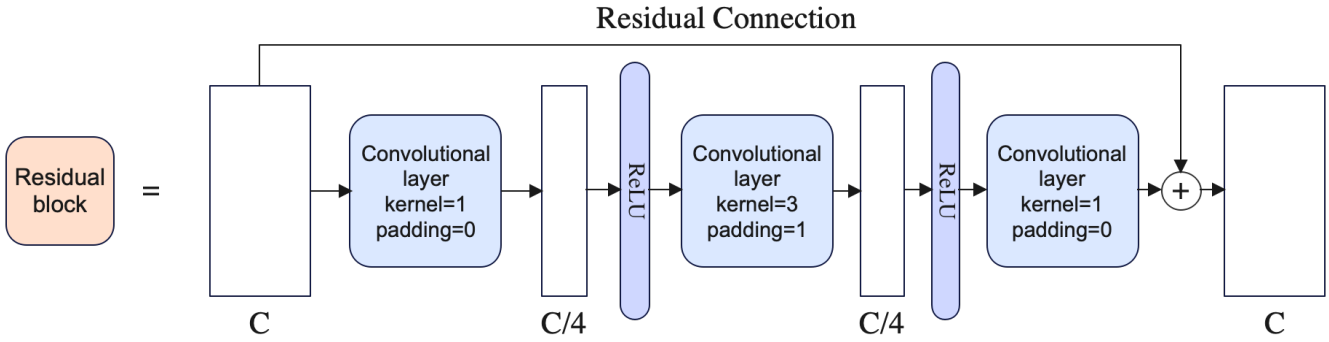

FIG. 12. Design of the residual block. The block consists of two pointwise convolutional layers corresponding to full connection layers in the channel direction and a convolutional layer with a kernel size of 3. The padding is adjusted so that the input and output tensors have the same shape. This structure is the same as the one used in Resnet. The presence of residual coupling allows learning to proceed without gradient loss, even as the depth of the neural network increases.

[6] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, Nature **550**, 354 (2017).
[7] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91** (2019), 10.1103/RevModPhys.91.045002.
[8] M. Raissi, P. Perdikaris, and G. Karniadakis, J.Comput.Phys. **378**, 686 (2019).
[9] R. Hong, P.-F. Zhou, B. Xi, J. Hu, A.-C. Ji, and S.-J. Ran, SciPost Phys. Core **4**, 022 (2021).
[10] A. Sehanobish, H. H. Corzo, O. Kara, and D. van Dijk, (2021), arXiv:2006.13297 [physics, physics:quant-ph, stat].
[11] J. Pu, J. Li, and Y. Chen, (2020), arXiv:2011.04949 [nlin].
[12] M. Nakajima, K. Tanaka, and T. Hashimoto, IEEE Trans. Neural Netw. Learning Syst. , 1 (2021).
[13] K. Mills, M. Spanner, and I. Tamblyn, Phys. Rev. A **96**, 042113 (2017).
[14] R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, Phys. Rev. Lett. **124**, 010508 (2020).
[15] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, in *International conference on machine learning* (PMLR, 2019) pp. 7354–7363.
[16] J. Izaac and J. Wang, *Computational Quantum Mechanics*, 1st ed. (Springer, New York, NY, 2019).
[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, arXiv:1810.04805 (2018).
[18] L. Wolfenstein, Phys. Rev. D **17**, 2369 (1978).
[19] S. Mikheyev and A. Y. Smirnov, Il Nuovo Cimento C **9**, 17 (1986).
[20] S. J. Parke, Phys. Rev. Lett. **57**, 1275 (1986).
[21] H. A. Bethe, Phys. Rev. Lett. **56**, 1305 (1986).