Probabilistic neural networks for improved analyses with phenomenological *R*-matrix

C. H. Kim,[1] K. Y. Chae,[1,*] M. S. Smith,[2] D. W. Bardayan,[3] C. R. Brune,[4] R. J. deBoer,[3] D. Lu,[5] and D. Odell[4]

[1]*Department of Physics, Sungkyunkwan University, Suwon 16419, Republic of Korea*
[2]*Physics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA*
[3]*Department of Physics and Astronomy, University of Notre Dame, Notre Dame, Indiana 46556, USA*
[4]*Department of Physics and Astronomy, Ohio University, Athens, Ohio 45701, USA*
[5]*Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, USA*

(Received 26 February 2024; revised 13 June 2024; accepted 2 October 2024; published 18 November 2024)

We present a method for measurement analyses based on probabilistic deep neural networks that provide several advantages over conventional analyses with phenomenological models. These include predicting physical quantities directly from data, the rapid generation of statistically robust uncertainties, and the ability to bypass some parameters that may induce ambiguities and complications in data analysis. As deep learning methods make predictions through "black boxes," the uncertainty quantification is typically challenging. We use a probabilistic framework that provides thorough uncertainty quantification and is straightforward to follow in practice. With the network architecture based on the Transformer, we demonstrate the current method for predicting nuclear resonance parameters from scattering data using the phenomenological *R*-matrix model.

DOI: 10.1103/PhysRevC.110.054609

I. INTRODUCTION

In modern physics, phenomenological models are routinely used to understand physical systems by fitting their predictions to measured data and extracting "best fit" values for their parameters [1–4]. To determine their numerous parameters, each of which has its own particular contribution to observables, χ^2 minimization or Bayesian inference is typically used [5,6]. In some cases, however, the quantification of uncertainties is not straightforward, and their calibrations are questionable [7,8]. Additionally, models may have some parameters that do not correspond to observable features in measurements but rather induce complications and ambiguities during the interpretation of measurement results [9,10].

Such challenges motivate the exploration of new data analysis approaches. Deep learning, with its ever-increasing capabilities, is currently one of the most promising alternatives to mitigate the difficulties of conventional analysis methods. Deep learning approaches utilize high-capacity deep neural networks with the flexibility to approximate a wide variety of models [11–14], making their adoption the norm for data analyses in many areas of physics [15–20]. While the inherent "black box" character of standard deep neural networks usually impedes the assessment of uncertainties, incorporating probabilistic frameworks enables uncertainty quantification using stochastic features [21–23], thereby broadening their utility for physics research.

This study introduces a general purpose, straightforward method based on a probabilistic deep learning framework known as deep ensembles that offers numerous advantages over data analyses with phenomenological models. We demonstrate that our trained deep learning model has the capability to reliably and rapidly extract features directly from data with statistically calibrated quantification of uncertainties. Furthermore, some phenomenological model parameters that may induce ambiguities in data interpretations can be bypassed by suitable training approaches.

We first describe a method to analyze data using a deep learning model based on a phenomenological physics model in Sec. II. We demonstrate the method on the well-known phenomenological *R*-matrix in Sec. III, showcasing its capability. In Sec. IV, we present the conclusion and possible future works. Fundamental concepts and mathematical backgrounds of (probabilistic) deep learning are given in the Appendix.

II. METHOD

Deep learning has a high capacity to handle complex physics tasks using a deep neural network. Here, we show how to use deep learning to replace existing phenomenological models. A key feature is that the deep learning model can be designed to use only a subset of information to make predictions, eliminating parameters that may cause complications.

Figure 1 shows the workflow of the method, where $\boldsymbol{\theta}$ represents parameters of phenomenological models rather than neural networks. First, to generate a training dataset various combinations of parameters required by the phenomenological model are sampled (Fig. 1). A parameter set includes parameters of interest $\boldsymbol{\theta}$ and, if present, any other extraneous parameters $\boldsymbol{\theta}'$ required by the model that may introduce ambiguities in the analysis. The phenomenological model predictions calculated using the sampled parameters can be processed with a noise function $\mathcal{N}$ to simulate the noise

*Contact author: kchae@skku.edu

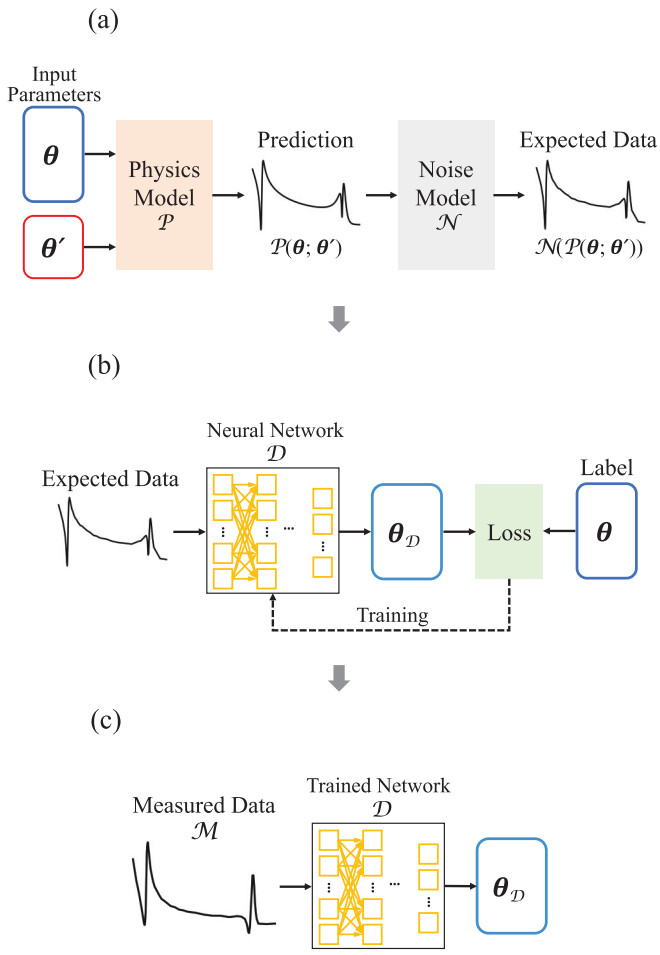

FIG. 1. Workflow of our method. (a) Training data generation using the physics model with sets of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. (b) Deep learning modeling with training data labeled only by the parameters of interest, $\boldsymbol{\theta}$. (c) Inference from the measurement. The trained model requires only measurement data to extract parameters at the inference stage.

present in experimental measurements, generating the "expected data."

Then, as shown in Fig. 1(b), a deep neural network $\mathcal{D}$ is trained on this dataset using the expected data as inputs and $\boldsymbol{\theta}$ as the labels. The key to eliminating unwanted parameters $\boldsymbol{\theta}'$ is to label the training data with *only* the parameters of interest $\boldsymbol{\theta}$ so that the deep learning model can extract these from the data without any knowledge of the unwanted parameters $\boldsymbol{\theta}'$. By training over data generated with a wide range of parameter values, we ensure that the deep learning model works on data with any parameter values.

After training, the model is used to predict the parameters directly from measured data $\mathcal{M}$, where $\boldsymbol{\theta}_{\mathcal{D}}$ represents the prediction [Fig. 1(c)]. The correct choices of $\boldsymbol{\theta}'$ are no longer required, as the model does not demand determinations of these parameters. This inference requires negligible execution time as the data only needs to pass forward through the model once.

Phenomenological models are normally used to extract parameters that have physical meaning or reproduce measurements for extrapolation purposes [24]. While Fig. 1 presents the method for the parameter extraction, the reproduction can be easily done by training the network to output the extrapolated expected data instead of the parameters.

This simple approach is applicable without specific restrictions to a wide variety of phenomenological physics models used to analyze measurement data. The physics model is only used to calculate training data; the rest of this approach only involves deep learning modeling. High-capacity neural networks with sufficient depth can learn patterns in training data from any physics model and then be used to analyze measured data instead of the original model. However, the deep learning model does not reconstruct the physics formalism to predict the targets, even though it is trained on the data from the physics model. The parameters in the network are simply fitted to the data, and the interpretation of these parameters is challenging due to the black box feature.

Conventional neural networks might not be suitable for replacing the phenomenological models, as the measurement analysis requires thorough uncertainty quantification. The use of probabilistic neural networks such as Bayesian neural networks (BNNs) in this method will give particular improvements and reliability in the analysis. See Appendix 5 for more details.

III. DEMONSTRATION ON PHENOMENOLOGICAL *R*-MATRIX

We demonstrated the method, as a proof of principle, on the phenomenological *R*-matrix that is widely used in nuclear physics [9,10,25–32]. The phenomenological *R*-matrix serves as a good test case because it contains numerous parameters, where some of them may induce complications during the analysis [9,10]. It is based on separating the particle interaction space into internal and external regions whose boundary is defined by a parameter known as the channel radius. The complex many-body nuclear interactions are present in the internal region and are described by the *R*-matrix, which is characterized by parameters of nuclear resonant states. From the *R*-matrix, expected observational data such as cross sections can be calculated. The parameters of nuclear states are determined by comparing the expected and experimental data. The extracted parameters provide the related nuclear properties, and the cross sections can be reproduced using the parameters for the purpose of extrapolation.

The fitting of physical parameters in the phenomenological *R*-matrix often depends on rather arbitrary choices of parameters that may not correspond to observable features in measurements [9,10]. The *R*-matrix theory requires an infinite number of nuclear states in the calculation. However, the truncation of the number of states is necessary for the phenomenological approach, as generally only a limited number of nuclear states are known. The truncated states are compensated by the so-called background poles, which are artificial states included to reproduce the effects of the truncated states. The strength of the background poles depends on the channel radius, which creates additional complications [10,25]. In practice, there are no strict physics rules to guide valid choices of the channel radius and background poles. Different

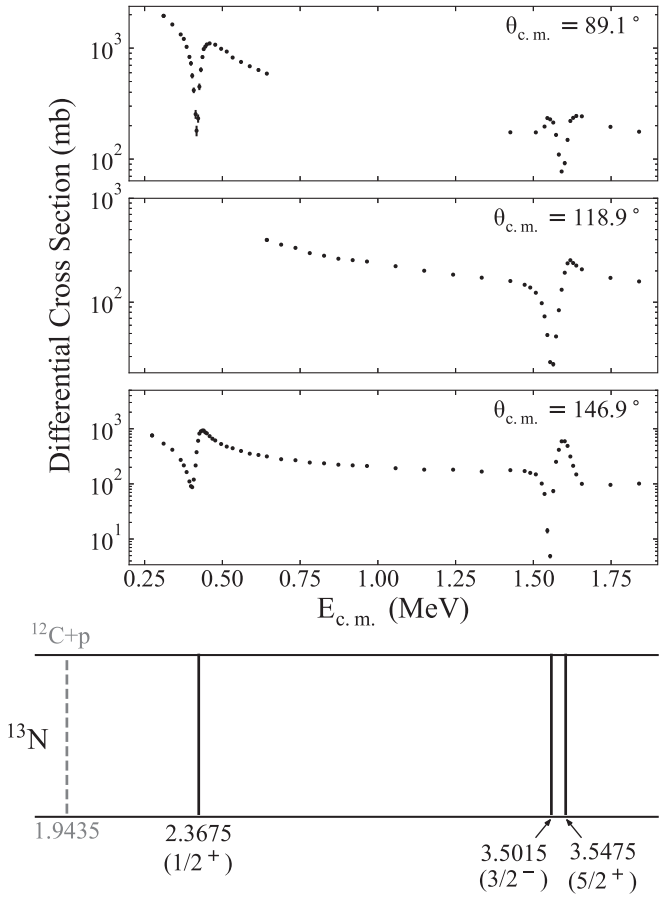

FIG. 2. The differential cross section spectra for the $^{12}C+p$ elastic scattering measurement of Ref. [33]. The data is taken from EXFOR [34]. The last subplot shows the corresponding energy levels of the compound nucleus ^{13}N, where the spin-parities are taken from [35].

choices often produce different fitting results because they can create different theoretical predictions. This feature can cause difficulties in the phenomenological analysis and uncertainties of the results, requiring sensitivity tests of fitting results to these parameters [9,10].

One of the major applications of the phenomenological R-matrix is determining resonance parameters (e.g., spins J, parities π, energies E, widths Γ) in nuclear reaction data. We built a deep learning model to find patterns of resonance parameters in reaction cross section spectra. Specifically, our goal was to extract the parameters of three resonance states from the well-known $^{12}C+p$ elastic scattering measurement of Ref. [33]. We also bypassed the need to determine the channel radius and background poles, eliminating ambiguities that these can introduce.

A. Data preparation

The target measurement of Ref. [33] contains differential cross section at three different angles and E_{cm} from 0.25 to 1.85 MeV (Fig. 2). This energy range includes three resonances that correspond to the first three excited states of ^{13}N. While properties of these states have been well constrained, in this demonstration we assumed a case where some of the properties were not determined to see if the deep learning model can predict these well [35]. The inputs to the model were the differential cross section spectra, and the labels were the corresponding resonance parameters.

TABLE I. The ranges of parameters used to generate training data. We covered various possible cases in practice. J^π were uniformly sampled from possible spins and parities for the given l, where l is the relative orbital angular momentum of the particle pair. In total, five BGPs are included. See text for more details.

Parameters	Data distribution
J_1^π	$\frac{1}{2}^+$ (fixed)
J_2^π	$l \leqslant 2$
J_3^π	$l = 2$
E_1 (MeV)	Normal (0.425, 0.003)
E_2 (MeV)	Normal (1.565, 0.010)[a]
E_3 (MeV)	Normal (1.610, 0.006)[a]
Γ_1 (keV)	Normal (30, 3)
Γ_2 (keV)	Normal (50, 15)
Γ_3 (keV)	Normal (60, 8)
ANC ($fm^{-1/2}$)	Normal (1.81, 0.07)
a_c (fm)	3–8
5 BGPs [J^π]	$l \leqslant 2$
5 BGPs [E]	Uniform (3, 10)
5 BGPs [Γ]	Uniform (0, Γ_W)[b]

[a]We set E of the second and third states to be separated more than 10 keV as the energy bin sizes of the measurement data in this region are $\geqslant$ 10 keV.
[b]Γ_W is the proton width calculated from $\Gamma_W = 2P\gamma_W^2 = 2P(3\hbar^2/2\mu a_c^2)$, where P is the penetrability, γ_W^2 is the Wigner limit, and μ is the reduced mass.

We first sampled possible sets of parameters for three resonances, along with random values of the channel radius (a_c) and random configurations of background poles (BGPs). Table I shows the distributions of R-matrix model parameters used to generate the training data. We assumed various cases in practice to demonstrate that any situation can be handled as if some of the parameters have been determined by previous studies; e.g., J^π of the first resonant state (J_1^π) has been fully determined as $1/2^+$, J_2^π has not been constrained, and J_3^π has been partly constrained. For E and Γ, we similarly used Gaussian distributions with different uncertainties.

The asymptotic normalization coefficient (ANC) for the subthreshold state (the ground state) was also included in the R-matrix calculation but not as one of the labels to simplify the demonstration. The range was taken from the previous study [28,36]. We set the range of the channel radius to 3–8 fm, which contains common values being used in typical R-matrix calculations. A level for each possible spin-parity was included as a background pole. The location and width of these levels were randomly determined within the given ranges in Table I.

Each set of parameters was used to calculate the corresponding cross section using the R-matrix code AZURE2 [28]. The target measurement data has differential cross section values on certain energy bin points at three angles [33].

The calculated cross section values on the same bin points at the three angles were taken as the input. As the locations of bin points and values of angles were fixed, they were null values in training and not included in the data. Therefore, the input data samples were three-dimensional tensors with the shape of $(3, n_{\text{bin}}, 1)$. 3 represents the number of measured angles, n_{bin} is the number of bin points, and 1 represents the value of the cross section. We added random noises in the calculated cross section values, assuming Gaussian noises to reflect the measurement errors. The standard deviations of the Gaussians were obtained from the measurement errors given by Ref. [33].

We set the logarithm of the differential cross sections as the data inputs to the model, which numerically mitigates the order of magnitude differences between cross section values. We also used $\ln(E)$ and $\ln(\Gamma)$ to label energies and widths. With Gaussian likelihoods, the model will eventually give the log-normal distributions for E and Γ predictions, which has some well-known advantages for representing positive-definite physical quantities [37]. J and π were separately encoded using the one-hot encoding (see Appendix 1) and handled by classification. We did not include the information of a_c and BGPs in the inputs or labels.

B. Model

We used the transformer architecture as the base for our model [38]. The transformer is currently one of the most effective deep learning architectures used in a wide variety of applications [13,39–41]. Its detailed description can be found in Ref. [38]. Here, we briefly summarize its main features. The transformer was originally constructed to handle sequence data for natural language processing. For translation purposes, the input and output are, for example, sentences from two different languages. The architecture is based on an encoder-decoder structure. Both encoder and decoder consist of multiple layers of multihead attentions, fully connected feed-forward networks, residual connections, and layer normalizations [42,43]. A multihead attention splits the incoming data and applies multiple attention functions known as scaled dot-product attentions to let the model attend to the data points at multiple aspects [38]. An encoder layer contains a multihead self-attention and feed-forward network. On the other hand, a decoder layer contains an additional multihead attention known as an encoder-decoder attention performing with the encoder output.

The attentions find critical relations between two sequence data, known as a query and key, and use them to update another sequence data, known as a value. Queries, keys, and values in the self-attentions originate from the input. On the other hand, in the encoder-decoder attentions, queries originate from the decoder input, and keys and values originate from the encoder output. The obtained relations are often called attention (score) matrices, and they present locations of data points critical for the model predictions.

We modified the original architecture to be suitable for our work. Figure 3 shows a sketch of the architecture. The inputs to the encoder and decoder are originally embedded and then passed to the positional encoding function to convert

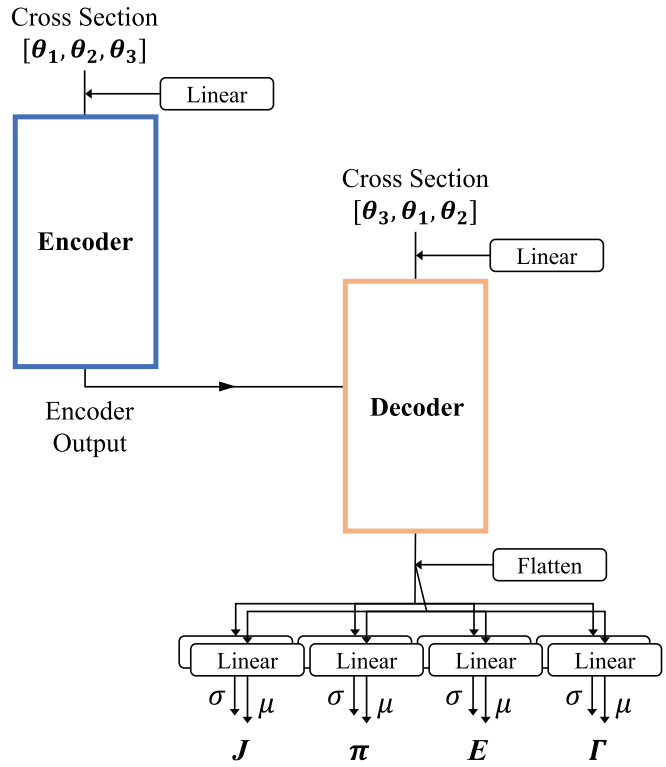

FIG. 3. Sketch of the transformer-based architecture for this study. The figure is adapted from Ref. [24].

the sentences to vectors and encode the positional information of each word [38]. In this study, we removed such functions as the inputs are not exactly sentences but spectra with continuous values. We simply replaced the embedding and positional encoding with a linear layer that projects a cross section point to a vector. After the first linear layer, the input shape $(3, n_{\text{bin}}, 1)$ was converted into $(3, n_{\text{bin}}, d_{\text{model}})$, where d_{model} is a hyperparameter of the transformer [38]. A cross section value, therefore, may correspond to a word in the natural language processing using the original transformer. Additionally, we used the Gaussian error linear units function as the activation function in the feed-forward networks of encoder and decoder layers [44].

Relations between the spectra of different angles are important in the analysis. The input data contain an extra dimension for the angles in addition to the two-dimensional cross section spectrum. However, the main operations in the transformer are matrix multiplications that perform on the last two dimensions of data. This feature limits the operations within a spectrum at each angle. For this reason, we gave the decoder input the same as the encoder input but shifted at the extra angle dimension, $[\boldsymbol{\theta_1}, \boldsymbol{\theta_2}, \boldsymbol{\theta_3}] \rightarrow [\boldsymbol{\theta_3}, \boldsymbol{\theta_1}, \boldsymbol{\theta_2}]$, where $\boldsymbol{\theta_i}$ represents a cross section spectrum at the ith angle. By doing so, the matrix multiplications in the encoder-decoder attentions can be done between the spectra at different angles. The decoder input does not require the look-ahead mask demanded by the original transformer, which uses target sequences as decoder inputs.

TABLE II. Hyperparameters in the model. We followed the notation of Ref. [38].

Hyperparameter	Value
d_{model}	256
d_{ff}	256
d_{q}, d_{k}, d_{v}	32
h	8
The number of encoder layers	2
The number of decoder layers	2

Model hyperparameters are written in Table II. We chose reasonable values for the hyperparameters based on several empirical test runs but did not fine tune their values. We found that the model performance was not highly sensitive to different hyperparameters or architectures. The output of the modified transformer was reshaped and passed to linear layers at the end of the model to return the four types of parameters, as shown in Fig. 3. Because of the different nature of our four parameters J, π, E, and Γ, we split the architecture output into four branches where two of them predict J and π and the others predict $\ln(E)$ and $\ln(\Gamma)$.

We used deep ensembles on top of this architecture. We found that $N_{\text{model}} = 5$ was sufficient to obtain well-calibrated uncertainties (see Sec. III C and Fig. 5). The final layer of the model generates four predictions, i.e., $\ln(E)$, $\ln(\Gamma)$, J, and π. For $\ln(E)$ or $\ln(\Gamma)$, it produces the mean and variance of the Gaussian likelihood. For the classification of J and π, we utilized the method presented by Ref. [45]: the logit vectors are sampled from a Gaussian distribution, with the mean and variance being those predicted by the model.

In the deep ensembles approach, the training procedure of each point-estimate model is similar to that of conventional neural networks (see Appendix 5). We performed the training for 4200 epochs with ≈21 000 training samples and ≈5000 validation samples. The batch size was 768 on 3 GPU workers. The number of samples could be freely increased at the expense of training time. We used an Adam optimizer with a weight decay of 10^{-9} [46]. The learning rate was 0.0002 with a scheduler that decreased the learning rate by a factor of 0.99 at every 6 epochs. We constantly monitored the mean errors of predictions and NLL. After the training, we tested the model with a relatively large number of samples, ≈30 000, enough to plot reliability diagrams with a sufficient number of samples in each bin.

C. Results

This section includes diverse tests of the proposed method and model on the test dataset, out-of-distribution samples, and target measurement. The predictive distributions using deep ensembles are obtained from the average of the multiple model predictions (see Appendix 5). Here, the multiple Gaussian likelihoods of $\ln(E)$ and $\ln(\Gamma)$ were first converted to log-normal distributions of E and Γ, and the mixture of these was approximated as another Gaussian distribution. J and π were obtained using the average of the softmax outputs from the multiple models.

TABLE III. Test results of our model. Model performance is presented with accuracy (for J and π) and mean errors (for E and Γ). The median values of the predictive distributions are taken to calculate the errors.

Parameter	Model performance
J_2	98.7%
π_2	100.0%
J_3	99.0%
E_1 (keV)	0.14 (0.03%)
E_2 (keV)	1.24 (0.08%)
E_3 (keV)	0.59 (0.04%)
Γ_1 (keV)	0.20 (0.67%)
Γ_2 (keV)	1.98 (4.39%)
Γ_3 (keV)	1.32 (2.23%)

Table III shows the accuracies and errors of the trained model on the test dataset. The accuracies on J^π reach ≈99%, and the mean errors on E and Γ are <≈2 keV. The errors are higher on the parameters that have wider ranges in the training data distribution, as the model faces more difficulties with parameters that cause more complicated variations. By using more training data or fine tuning model architecture, this performance can be further improved if needed.

We also tested the model performance using the values of eliminated parameters used to calculate the cross sections during the training data generation. Figure 4 shows accuracies and normalized errors of predictions for cross section spectra calculated with different values of eliminated parameters. The figures for the channel radius and first background pole, in terms of energy, are shown as representatives. The model performances are fairly even—less than ≈2% accuracy and ≈1 keV (35%) error variations for classification and regression, respectively—on these parameters. This demonstrates that the deep learning model performs well on data that originated from any values of the bypassed parameters. Nevertheless, certain tendencies can be seen, e.g., higher errors for the lower BGP energy, as the BGP will have a greater impact on the measured energy range.

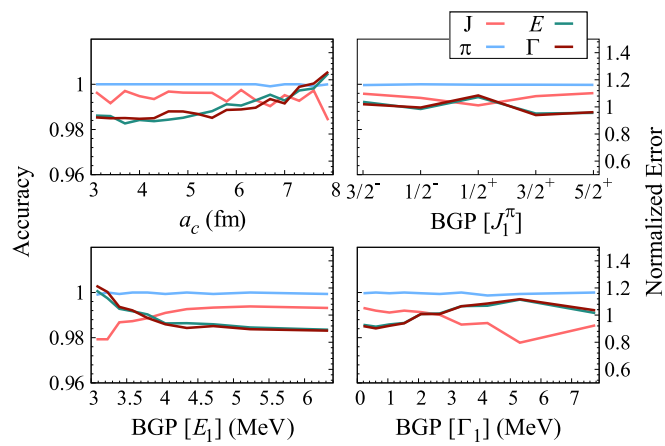

FIG. 4. Mean accuracies and errors of the model predictions as a function of bypassed parameters. See text for more details.

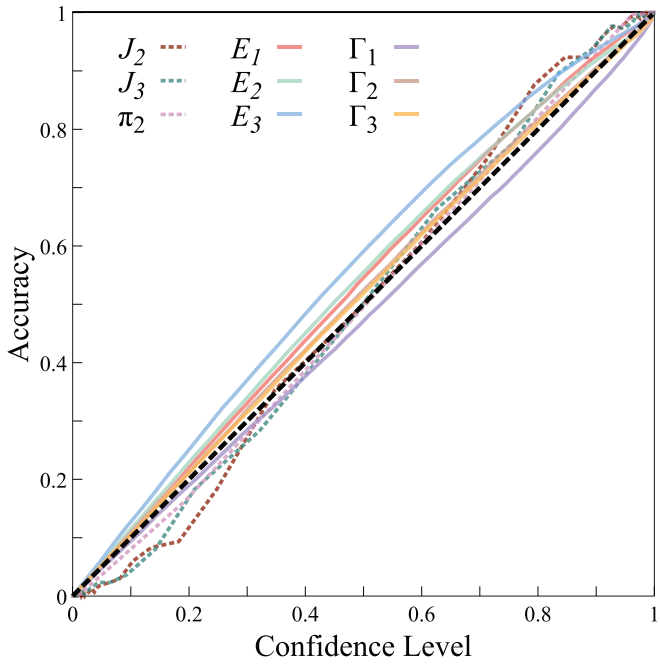

FIG. 5. Reliability diagram of the current model. Each curve shows the estimated accuracy (fraction of true data points included in the confidence range) as a function of the corresponding confidence level.

Figure 5 shows the reliability diagram which is the standard method to evaluate the calibration of uncertainty (see Appendix 2). For discrete parameters (J and π), the confidence level is the predicted probability from the softmax function, and the accuracy is the fraction of correct predictions. For continuous parameters (E and Γ), the confidence level is calculated from the range surrounding the median value, and the accuracy is the fraction of true data points included in the corresponding range. The dashed black line shows the ideal case where the confidence level is equal to the accuracy. It is clear that our probabilistic model achieved a performance close to this ideal case.

Model predictions on data samples out of the training data distribution are normally unreliable. It is desired to have a model that can detect such out-of-distribution data samples through uncertainty quantification. We made a set of samples for various cases shown in Table IV to test if the model can detect such samples. As shown in Fig. 6, we found that the model outputs large (epistemic) uncertainties compared to the case of the test dataset. See Appendix 2 for more discussion on out-of-distribution data.

We derived the resonance parameters from the target measurement data of Ref. [33] using the model. The inferences for the measurement can be easily made by simply passing the cross section spectra to the model. Tables V and VI show the model results on the measurement data. The inferences of the spin-parities of the second and third states are, at ≈100% probability, $J_2 = 3/2$, $\pi_2 = -$, and $J_3 = 5/2$, in agreement with the literature [35]. The inferences of E and Γ are mostly consistent with those of the previous R-matrix studies [9,28,33], demonstrating the effectiveness of our model. The

TABLE IV. Various cases for the out-of-distribution data for a test. The training dataset was composed of the ^{12}C $+p$ reactions with three resonances. The input spectra for case 3 were constant lines including no resonance.

J^π	$E_{\rm res}$	Γ
Case 1: ^{12}C $+p$ reaction with 4 resonances		
$1/2^+$	0.4244 MeV	34.3 keV
$3/2^-$	1.5590 MeV	55.0 keV
$3/2^+$	1.5765 MeV	30.0 keV
$5/2^+$	1.6029 MeV	50.3 keV
Case 2: ^{6}Li $+p$ reaction		
$1/2^+$	0.4244 MeV	34.3 keV
$3/2^-$	1.5590 MeV	55.0 keV
$5/2^+$	1.6029 MeV	50.3 keV
Case 3: Constant line without any resonance		
n/a[a]	n/a	n/a

[a]Not available.

previous analyses used different channel radius values and background pole configurations, which might influence their results; e.g., Ref. [9] tested the sensitivity of the fitting results to the a_c, varying the values of a_c. On the other hand, our deep learning model did not require any use of these extraneous parameters, as it is trained to make predictions without any information about these.

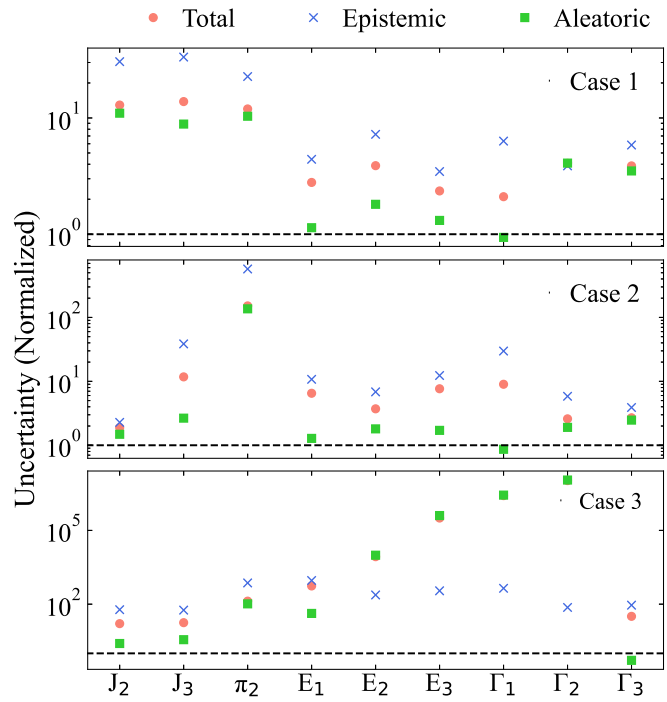

FIG. 6. Uncertainties of model predictions for the out-of-distribution samples in Table IV. They are normalized to the average uncertainties of model predictions on the test dataset. The large uncertainties indicate that the model has not been trained on such types of data samples.

TABLE V. Inferences of J and π for the target measurement data. Values in the parentheses are the probabilities presented by the model.

Parameter	Current model	Ajzenberg-Selove [35]
J_2	3/2	3/2
	(1/2: 1.06×10^{-4})	
	(3/2: 9.99×10^{-1})	
	(5/2: 1.81×10^{-4})	
J_3	5/2	5/2
	(3/2: 8.27×10^{-6})	
	(5/2: 9.99×10^{-1})	
π_2		
	(+: 2.05×10^{-4})	
	(−: 9.99×10^{-1})	

Figure 7 shows the attention matrices in a self-attention and encoder-decoder attention of our model. Although a detailed interpretation of the matrices is challenging, they show important relations between the two input sequences (two cross section spectra). Higher values in the matrices indicate that more attention (weights) was paid to the corresponding data points during the calculation. We can find the model gave more weight not only to the peaks but also to the valleys of the spectra to find values of the resonance parameters.

D. Implications

The model of this demonstration is specialized for the $^{12}C+p$ measurement of Ref. [33]. Generalized models that address numerous similar problems can also be built with a larger model and dataset, e.g., a model flexible to the number of resonances. More practically, a model pretrained on a relatively large dataset can be utilized [39,47]. It only needs to be slightly adjusted to handle a specific task. Compared to typical deep learning applications that require piecemeal hand labeling, here, data preparation is relatively effortless, significantly easing the model training process. Still, running some physics models may require large computational costs which will undermine such advantages.

A certain finite range should be defined for training data, as creating data with a range $(-\inf, \inf)$ is challenging. If the ranges for training data are difficult to set, one can first try conventional fitting to find valid ranges. Still, such limitation also exists in the practice of conventional fitting, as it is not practically possible to try an infinite range of parameters. For example, a reaction with a high l value is largely suppressed by the centrifugal barrier, which allows an assumption of a l value less than a particular number.

We also note that the training data distribution does not play a role of prior in Bayesian methods, even though both may have the common feature of representing the known information. The data distribution does not have a direct mathematical effect as a prior does [24]. For example, if the prior follows a Gaussian, it will directly affect the determination of the posterior through Bayes's theorem. On the other hand, if the training data distribution is Gaussian, the model will be fitted on an imbalanced dataset, which may result in an imbalanced performance, i.e., better near the mean as the model is more trained on that location.

In cases where the values of the eliminated parameters in the physics model are difficult to obtain, the performance of the deep learning model can exceed that of the conventional fitting with the physics model. If the eliminated parameters could (by some means) be correctly specified, then the performances of the deep learning model and conventional fitting are comparable. The deep learning model can accurately identify distinctive features in the data, that arise from variations of labels in training data. In cases where variations of the eliminated parameters can interrupt the model from finding such features, the predictions may have high uncertainties with diminished performance [48,49].

Although it is challenging to make unambiguous quantitative one-on-one comparisons with conventional optimization methods such as χ^2 methods or Bayesian inference, there are several advantages of the probabilistic neural network-based method. First, once a deep learning model is trained, the

TABLE VI. Inferences of E and Γ for the target measurement data along with the previous results. For the current model, the median values of the distributions are presented with the standard deviations.

Parameter	Current model	Ref. [33][a]	Ref. [28][b]	Ref. [9][a]	Ref. [9][a]	Ref. [9][a]
E_1 (MeV)	0.4244(2)	0.424	0.426(3)	0.427	0.427	0.427
E_2 (MeV)	1.5590(12)	1.558	1.556(1)	1.560	1.559	1.558
E_3 (MeV)	1.6029(8)	1.604	1.602(2)	1.603	1.604	1.606
Γ_1 (keV)	34.3(3)	33	34.1(8)	33.8	32.9	30.9
Γ_2 (keV)	55.0(8)	55	57.9(17)	51.4	51.4	51.3
Γ_3 (keV)	50.3(7)	50	48.3(19)	48.1	48.1	47.8
a_c (fm)	n/a	4.0	3.4	4.0	5.0	6.0
Background poles	n/a	$3/2^+$ state[c]	[d]	[d]	[d]	[d]

[a]The uncertainties are not presented in the papers.
[b]The capture reaction data was also used in the R-matrix fitting.
[c]While it was not explicitly stated as a background pole, the experimentally known $3/2^+$ level at $E = 5.860$ MeV with $\Gamma = 1400$ keV was included in the calculation in Ref. [33].
[d]No background pole was considered.

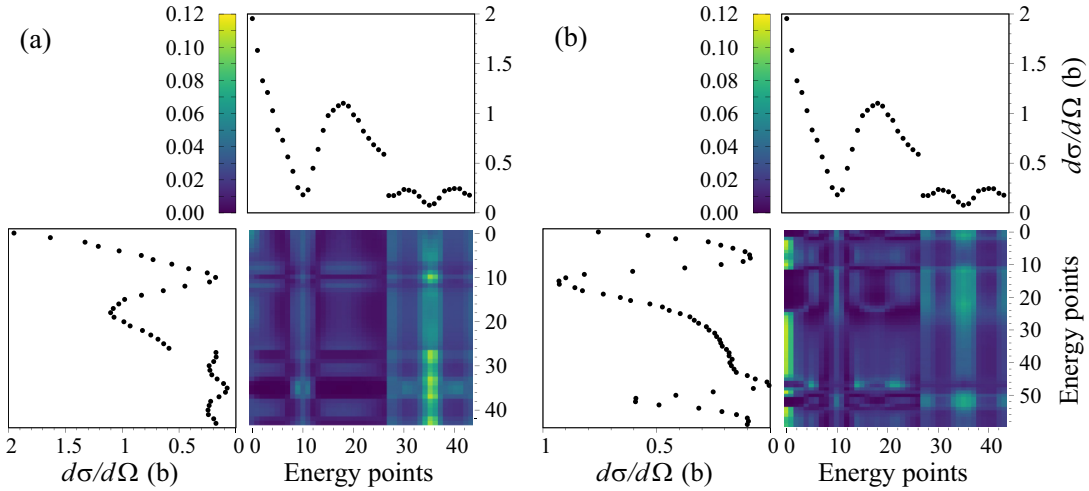

FIG. 7. Examples of the attention matrices when the target measurement data is passed as the input. (a) An attention matrix from a self-attention in the decoder. (b) An attention matrix from an encoder-decoder attention. The top and left plots from each matrix show the corresponding input data. See the text for the explanation of attentions.

inference calculation is normally very fast, with negligible computational times [17,50]. The model for the phenomenological R-matrix took ≈0.8 seconds with 1 GPU worker to estimate the nine parameters and corresponding uncertainties from the target measurement. Second, making probabilistic predictions about discrete quantities is straightforward using a classification approach, either with softmax functions or by defining specific distributions suitable for the likelihoods of the quantities. Third, the quality of the uncertainties in the model predictions can be rigorously tested through an analysis of a subset of the data (i.e., a test dataset) using well-developed measures such as a reliability diagram [7,22,51,52]. Robust methods also exist for cases where the uncertainty is not properly calibrated [51–53].

IV. FUTURE WORK

In this study, we introduce a method to analyze measurement data using deep probabilistic neural networks, which offers numerous advantages over traditional approaches with phenomenological models. The introduction of deep ensembles combines the abilities of neural networks to generate solutions of complex phenomena with an easy-to-use, statistically robust mechanism to quantify uncertainties of model predictions. We also showed that we can build a deep learning model trained to perform accurate inferences from data without specific determination of a subset of phenomenological model parameters.

The phenomenological R-matrix has been intensively used in nuclear physics to estimate resonance parameters and reproduce cross sections. Still, in practice, significant time and effort are required for the R-matrix fitting, as numerous parameters are involved, including the channel radius and background levels. We hope the current method can mitigate such issues by utilizing the capacity of deep neural networks.

We implemented the transformer for the architecture and deep ensembles for the probabilistic framework in the demonstration. The transformer has significant potential for implementation in numerous physics tasks, given its widespread utilization across diverse scientific disciplines [13,39–41]. Not only deep ensembles but also a method known as Monte Carlo dropout is straightforward to use even for nonexperts [54]. Such methods can be extensively used for deep learning applications in physics with well-quantified uncertainties.

We anticipate that this approach can be widely used for data analyses in various fields and to boost the development of reliable deep learning applications.

ACKNOWLEDGMENTS

We thank D. Phillips for helpful discussions on this study. This work was supported by National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT), Grants No. RS-2024-00338255 and No. 2020R1A2C1005981. This work was also supported in part by the Institute for Basic Science, Grant No. IBS-R031-D1; by the National Science Foundation, Grant No. NSF PHY-2011890; by the U.S. Department of Energy (DOE), Office of Science, Office of Nuclear Physics, Grants No. DE-AC05-00OR22725 and No. DE-FG02-88ER40387; and by the U.S. DOE, National Nuclear Security Administration, Grant No. DE-NA0004065. Computational works for this research were performed on the data analysis hub Olaf in the IBS Research Solution Center.

APPENDIX

Deep learning has already been extensively utilized in physics research [15–20]. Nevertheless, due to the quick evolution of the technique, misunderstandings of its concept and modeling still prevail. In this Appendix, we present the fundamentals of deep learning, from the basics of machine learning to probabilistic neural networks, to give a better understanding of the current study and future applications.

1. Basics of machine learning

Machine learning models generally refer to computer programs that learn to solve some tasks from experiences [55], where deep learning is a subset of machine learning. Specifically, in machine learning, a model is fitted on refined data, which is known as "training," to handle data in an actual environment.

Machine learning models are generally categorized into generative and discriminative. While generative models find a joint distribution over given variables, discriminative models give predictions for given observations [56]. Additionally, there are three learning methods: supervised, unsupervised, and reinforcement learning [57]. In supervised learning, the training data for machine learning carry the corresponding answers (labels); in unsupervised learning, the model learns on unlabeled data. Reinforcement learning uses rewards and punishments to build an agent that can make decisions to accomplish a task. In this study, we use supervised learning to make discriminative models.

For supervised learning, tasks are normally handled differently depending on labels. For regression tasks, such as predicting energies of incoming particles, the data labels are composed of continuous values; for classification tasks, such as identifying a track in a detector as a proton or alpha particle, the labels are discrete. Labels in classification are typically encoded to numerical values using one-hot encoding [57]: for example, if the label is "proton" or "neutron," they can be converted to [1, 0] or [0, 1], respectively.

Data are the essence of any machine learning task, and it usually requires significant effort to obtain a dataset that appropriately spans the relevant parameter space. Once obtained, training data are normally divided into three subsets: training, validation, and test datasets. The training dataset is used to fit the model parameters, and the validation and test datasets are used for statistical evaluation The model performance is evaluated using the validation dataset during the training and hyperparameter tuning processes; the test dataset is used for the final evaluation of the fitted model.

2. Uncertainty in machine learning

The reliability of any model prediction can be quantified by the corresponding uncertainty. This is essential for machine learning applications in physics research, especially when the model directly predicts physical quantities.

Any data-based model, even those constructed using conventional methods, only guarantees its functionality within the distribution from which the data was sampled. Therefore, in actual practice, identifying a sample outside of this distribution is important. It is preferred to have an uncertainty quantification method that can detect such cases [22,52].

Uncertainty in machine learning is normally categorized into epistemic and aleatoric, representing uncertainties caused by the model and data, respectively [48,49]. The epistemic uncertainties are related to the ignorance of the model due to a lack of data; the aleatoric uncertainties are related to the intrinsic uncertainties of the data itself. The disentanglement of these two could be useful in machine learning engineering, as it can guide how to reduce uncertainties of model predictions

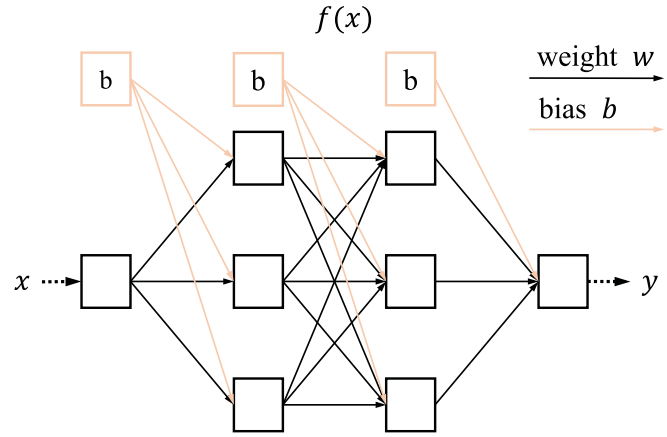

FIG. 8. Sketch of simple neural network. More layers and nodes make the network deeper. The figure is adapted from Ref. [24].

[49]. Additionally, the size of epistemic uncertainty can be used to detect the out-of-distribution samples, as it can show the ignorance of the model on the samples [22,52].

It is critical to quantify and calibrate uncertainties to address issues of reliability of deep learning for physics applications [24]. While the intrinsic complications of deep neural networks can make uncertainty quantification challenging [21,22], there are some well-developed methods, centered around the Bayesian neural networks (see Appendix 5).

The calibration of the quantified uncertainties can be statistically tested on a subset of the dataset (test dataset). In an ideal case, the frequency of a data point included in a confidence range should equal the corresponding confidence level. For example, if the model prediction is presented in a Gaussian distribution, the true data point should be located in the range $(\mu - \sigma, \mu + \sigma)$ at $\approx$68% probability. One representative metric is known as the reliability diagram [22,51]. It shows the frequency, or estimated accuracy, as a function of confidence level. See Sec. III C and Fig. 5 for more details.

3. Deep neural networks

Neural networks are the core of deep learning. One of the simplest conventional models is $y = f(x) = ax + b$. This function relates the input x and the target y using its model parameters a and b fitted to some linear data. It is fundamentally the same in deep learning, except that the function becomes a neural network with a lot of parameters depending on the number of layers and nodes. The most standard neural network is a densely connected network that has some layers connected to each other (Fig. 8). Each layer contains model parameters known as weights $\boldsymbol{w}$ and biases $\boldsymbol{b}$, acting as an operation $\boldsymbol{wx} + \boldsymbol{b}$ [57], and specialized activation functions enable the modeling of nonlinear systems [23].

Neural nets are said to be deep if they have many layers and a lot of parameters are entangled in the network. This feature gives a high capacity that can handle various tasks, compared to conventional models that are typically specialized to a particular subset of tasks (e.g., $y = ax + b$ for linear data). However, simultaneously, such structures make it difficult to

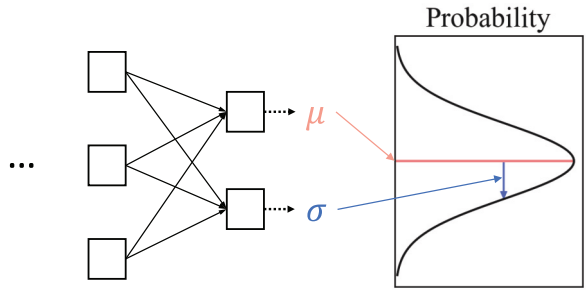

FIG. 9. The last layer can be designed to output mean and sigma for a Gaussian likelihood. The figure is adapted from Ref. [24].

interpret the roles of any particular model parameters, instead functioning as a "black box" [22].

The model performance frequently depends highly on the architecture of the neural network. Numerous effective architectures have been developed for various tasks, such as convolutional neural networks, recurrent neural networks, transformers, and more [57]. Currently, the transformers and their variations have demonstrated capabilities for complex modeling in a variety of fields, well beyond their initial development for natural language processing [38–40]. See Sec. III B for more details.

The difference in treatments for regression and classification in deep learning is at the last layer of the network. For regression, the last layer directly outputs the predictions (labels); for classification, a component of the output is normally interpreted as a probability for a possible class. In this latter case, the output is passed to a softmax function that converts it to a probability vector. The softmax is defined as

$$\text{Softmax}(z_c) = \frac{e^{z_c}}{\sum_{c'=1}^{N_{\text{class}}} e^{z_{c'}}}, \tag{A1}$$

which looks similar to the Boltzmann probability distribution [24]. z_c is known as a logit, which is an element of the output, and N_c is the number of classes.

4. Model training

The optimization of the deep learning model is normally based on the maximum likelihood estimation, the same as conventional model optimization approaches such as the χ^2 method [23,58]. This is typically done by minimizing the negative log-likelihood. For regression, if we assume a Gaussian likelihood, the negative log-likelihood (NLL) for a data sample becomes

$$-\ln[p(y|x,\boldsymbol{\theta})] = -\ln\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-y_p)^2}{2\sigma^2}}\right) \tag{A2}$$

$$= \frac{1}{2}\frac{(y-y_p)^2}{\sigma^2} + \ln(\sqrt{2\pi}\sigma), \tag{A3}$$

where y_p is the model prediction and $p(y|x,\boldsymbol{\theta})$ is the likelihood for the given input x, output y (which are often tensors), and parameters $\boldsymbol{\theta}$. For a constant σ, the NLL follows the mean squared error, as constant terms can be safely dropped during minimization. Otherwise, as shown in Fig. 9, the last layer of the network can output the mean and standard deviation that represents the data uncertainty (aleatoric) [45,59].

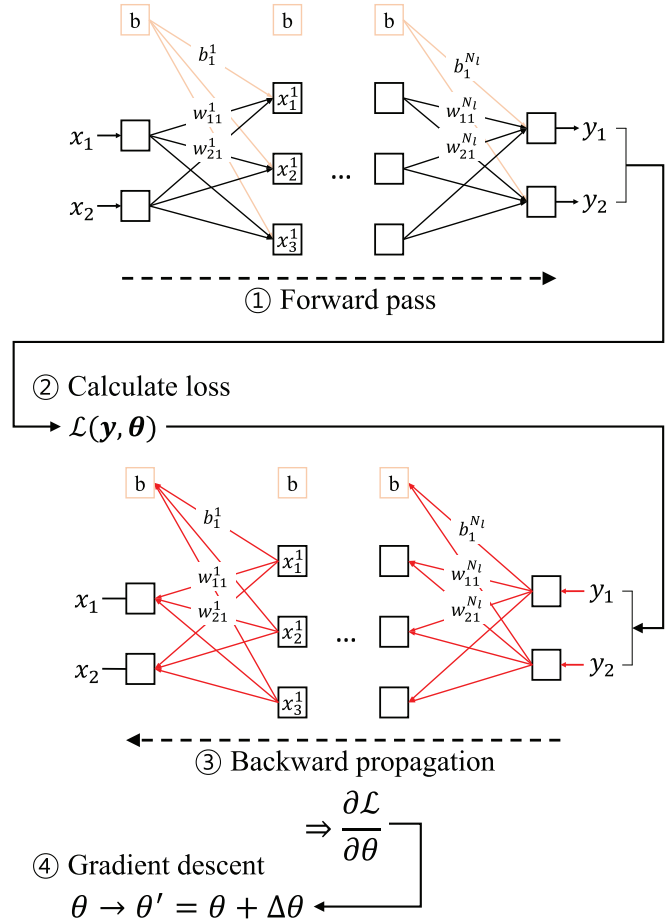

FIG. 10. Sketch of a typical process for fitting a deep neural network. The figure is adapted from Ref. [24].

For classification, the likelihood is assumed to follow the softmax $p(y=c|x,\boldsymbol{\theta}) = \text{Softmax}(z_c)$. As in regression, the logit z can be replaced by a Gaussian distribution, where its mean and variance are predicted by the model [45]. This approach showed effective results regarding accuracy and uncertainty quantification [45,60].

To reduce the NLL, the gradient descent method is normally used [23]. This method uses the gradient of the loss function (NLL) with respect to the model parameters to determine the direction for adjusting model parameters. However, calculating the gradient is not straightforward because of the complicated structure of the network. Typically, an algorithm known as the backpropagation that utilizes the chain rule is used to obtain the gradients [23].

Figure 10 summarizes a typical training process in deep learning. First, the data are passed to the network to calculate the loss function with the labels. Second, the gradient is calculated using the backpropagation. Finally, the model parameters are adjusted toward the negative direction of the gradient. The detailed method for parameter updates is controlled by the so-called optimizer [57]. These steps are iterated until the local minimum of the loss function is obtained.

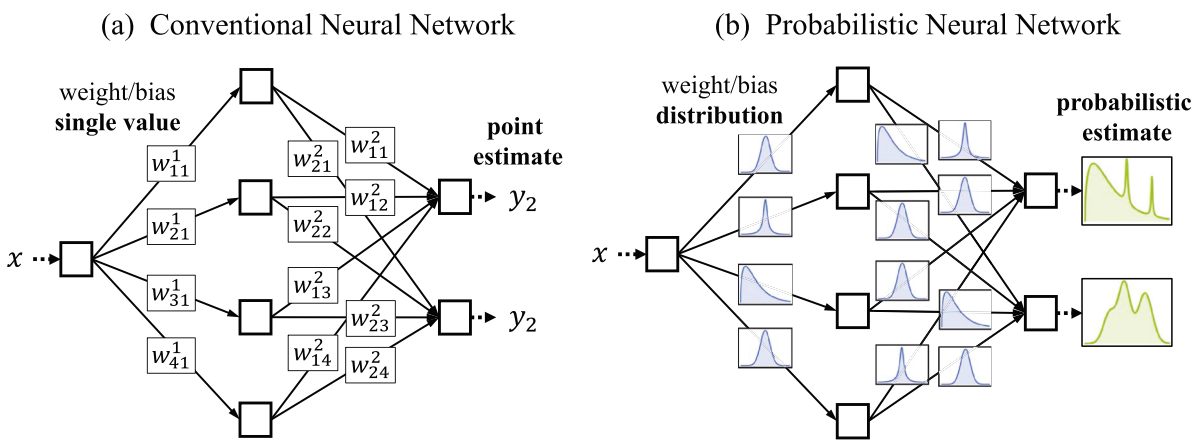

FIG. 11. Sketch for comparison between (a) point-estimate and (b) probabilistic neural networks.

One of the challenges in model training is the case known as overfitting, where the model is not well generalized outside of the training dataset [57]. Deep neural networks are easily specialized to the data used for fitting since they normally contain a large number of parameters. Overfitting can be detected from the difference between the model performances on the training and validation (test) datasets, as the latter is not used for optimization.

Treatments to prevent overfitting by restricting the complexities of models are known as regularization [57]. Regularization typically adds additional terms in the loss function to drive the optimization process to favor a simpler model. This can be seen as adding a prior to the likelihood, resulting in the maximum a posterior estimation [22,23]. One of the representative methods is weight decay, also known as $L2$ regularization [23]. This gives penalties for large parameter values by adding a term, $||\boldsymbol{\theta}||$, to the loss function, where λ controls the size of the penalty. This additional term can be seen as a Gaussian prior $p(\boldsymbol{\theta})$:

$$-\ln[p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{\theta})] \tag{A4}$$

$$= -\ln[p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})] - \ln[\text{Normal}(\boldsymbol{\theta}|0,1/2\lambda)] \tag{A5}$$

$$= -\ln[p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})] + \lambda||\boldsymbol{\theta}||^2 + \ln(\sqrt{2\pi}/2\lambda), \tag{A6}$$

$$\mathcal{L} = -\ln[p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})] + \lambda||\boldsymbol{\theta}||^2. \tag{A7}$$

The $||\boldsymbol{\theta}||$ term constrains the parameter values, as the Gaussian gives small weights far from the mean. λ controls the width of the Gaussian and, therefore, the range of the parameter values.

5. Bayesian deep learning

To be reliable, any model prediction should be accompanied by its corresponding uncertainty. Figure 11 shows the difference between conventional and probabilistic neural networks. Conventional neural networks typically only give a single value, known as a point estimate, for a prediction without the corresponding uncertainty. The black box feature prevents the interpretation of the parameters and, as a result, poses challenges for uncertainty quantification [22]. On the other hand, probabilistic neural networks can output predictions as probabilities or distributions. Particularly, Bayesian deep learning has been developed as the core method for uncertainty quantification in deep learning [21,59]. In this section, we introduce the concept of Bayesian neural networks (BNNs) and a representative technique known as deep ensembles [61].

Parameters $\boldsymbol{\theta}$ in BNNs are distributions or sampled from distributions determined using Bayes's theorem [22]:

$$p(\boldsymbol{\theta}|\boldsymbol{D}_x,\boldsymbol{D}_y) = \frac{p(\boldsymbol{D}_y|\boldsymbol{D}_x,\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{D}_y|\boldsymbol{D}_x)}, \tag{A8}$$

where $\boldsymbol{D}_x$ and $\boldsymbol{D}_y$ are inputs and labels of the training dataset. Training of BNNs is (Bayesian) inference to obtain the posterior $p(\boldsymbol{\theta}|\boldsymbol{D})$, where $\boldsymbol{D}$ represents the training dataset; in contract, the training of conventional neural networks is optimization [62].

A key feature of Bayesian deep learning is the marginalization over the model parameters using the posterior, erasing the dependency of predictions on the model parameters [62]:

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{D}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}')p(\boldsymbol{\theta}'|\boldsymbol{D})d\boldsymbol{\theta}'. \tag{A9}$$

$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{D})$ is the final prediction, known as the predictive distribution. Equation (A9) shows the way to quantify uncertainties in Bayesian deep learning. Bayesian deep learning uses every possible parameter set to determine the prediction, while a point-estimate model uses a single parameter set. The model (epistemic) and data (aleatoric) uncertainties are considered in the posterior $p(\boldsymbol{\theta}'|\boldsymbol{D})$ and likelihood $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}')$, respectively [22]. The likelihood can be estimated from the network outputs (see Appendix 4).

Normally, Monte Carlo methods are used for Bayesian inference [6]. However, due to a large number of parameters, Monte Carlo methods are computationally challenging for deep neural networks and are limited to use in very small networks, which may be insufficient to model complex physics phenomena [22]. For this reason, variational inference is critically used in Bayesian deep learning [59]. Variational inference is a method to approximate the posterior with a well-known distribution [63]. Additionally, many ideas, such as deep ensembles, have been developed to implement Bayesian deep learning easily [22,54,61].

While deep ensembles were first proposed as an alternative method to BNNs, they can be interpreted as another approach

to Bayesian deep learning [22,62]. Deep ensembles have been validated as a straightforward and highly effective method [52,61]. Multiple point-estimate models are first trained using NLL and random initial model parameters. The predictions from these models are then combined to get the predictive distribution:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{N_{\text{model}}} \sum_{n=1}^{N_{\text{model}}} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}_n), \tag{A10}$$

where N_{model} is the number of point-estimate models. This expression can be obtained from Eq. (A9) if the posterior follows a mixture of multiple delta functions: $p(\boldsymbol{\theta}|\boldsymbol{D}) \approx (1/N_{\text{model}}) \sum_{n=1}^{N_{\text{model}}} \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_n)$. Therefore, the idea of deep ensembles can be seen as the marginalization in Bayesian deep learning using variational inference with a mixture of delta functions as the approximate distribution [22,62].

In practice, deep ensembles can be done by simply training multiple conventional neural networks with different initial parameters. For regression with Gaussian likelihoods, the predictive distribution is a mixture of Gaussian distributions, which can be further approximated as a Gaussian [61]. For classification, the prediction will be the average of the softmax outputs from the models. Suitable N_{model} can be determined through statistical examinations of uncertainty calibration (see Appendix 2).

[1] C. Cutler and É. E. Flanagan, Gravitational waves from merging compact binaries: How accurately can one extract the binary's parameters from the inspiral waveform? Phys. Rev. D **49**, 2658 (1994).

[2] P. Möller, A. J. Sierk, T. Ichikawa, and H. Sagawa, Nuclear ground-state masses and deformations: FRDM(2012), At. Data Nucl. Data Tables **109-110**, 1 (2016).

[3] H. A. Wiltsche and P. Berghofer, *Phenomenological Approaches to Physics* (Springer, Berlin, 2020).

[4] D. Everett, W. Ke, J. F. Paquet, G. Vujanovic, S. A. Bass, L. Du, C. Gale, M. Heffernan, U. Heinz, D. Liyanage, M. Luzum, A. Majumder, M. McNelis, C. Shen, Y. Xu, A. Angerami, S. Cao, Y. Chen, J. Coleman, L. Cunqueiro *et al.*, Phenomenological constraints on the transport properties of QCD matter with data-driven model averaging, Phys. Rev. Lett. **126**, 242301 (2021).

[5] R. B. D'Agostino and M. A. Stephens, *Goodness-of-Fit Techniques* (Marcel Dekker, New York, 1986).

[6] U. von Toussaint, Bayesian inference in physics, Rev. Mod. Phys. **83**, 943 (2011).

[7] A. P. Dawid, The well-calibrated Bayesian, J. Am. Stat. Assoc. **77**, 605 (1982).

[8] D. L. Smith, S. A. Badikov, E. V. Gai, S.-Y. Oh, T. Kawano, N. M. Larson, and V. G. Pronyaev, *Perspectives on Peelle's Pertinent Puzzle* (IAEA, Vienna, 2007).

[9] P. Descouvemont and D. Baye, The *R*-matrix theory, Rep. Prog. Phys. **73**, 036301 (2010).

[10] R. J. deBoer, J. Görres, M. Wiescher, R. E. Azuma, A. Best, C. R. Brune, C. E. Fields, S. Jones, M. Pignatari, D. Sayre, K. Smith, F. X. Timmes, and E. Uberseder, The $^{12}C(\alpha, \gamma)^{16}O$ reaction and its implications for stellar helium burning, Rev. Mod. Phys. **89**, 035007 (2017).

[11] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, Deep physical neural networks trained with backpropagation, Nature (London) **601**, 549 (2022).

[12] F. Ashtiani, A. J. Geers, and F. Aflatouni, An on-chip photonic deep neural network for image classification, Nature (London) **606**, 501 (2022).

[13] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back *et al.*, Highly accurate protein structure prediction with AlphaFold, Nature (London) **596**, 583 (2021).

[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. **378**, 686 (2019).

[15] B. S. Rem, N. Käming, M. Tarnowski, L. Asteria, N. Fläschner, C. Becker, K. Sengstock, and C. Weitenberg, Identifying quantum phase transitions using artificial neural networks on experimental data, Nat. Phys. **15**, 917 (2019).

[16] A. Seif, M. Hafezi, and C. Jarzynski, Machine learning the thermodynamic arrow of time, Nat. Phys. **17**, 105 (2021).

[17] H. Gabbard, C. Messenger, I. S. Heng, F. Tonolini, and R. Murray-Smith, Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy, Nat. Phys. **18**, 112 (2022).

[18] A. Boehnlein, M. Diefenthaler, N. Sato, M. Schram, V. Ziegler, C. Fanelli, M. Hjorth-Jensen, T. Horn, M. P. Kuchera, D. Lee, W. Nazarewicz, P. Ostroumov, K. Orginos, A. Poon, X.-N. Wang, A. Scheinker, M. S. Smith, and L.-G. Pang, Colloquium: Machine learning in nuclear physics, Rev. Mod. Phys. **94**, 031003 (2022).

[19] C. H. Kim, S. Ahn, K. Y. Chae, J. Hooker, and G. V. Rogachev, Noise signal identification in time projection chamber data using deep learning model, Nucl. Instrum. Methods Phys. Res., Sect. A **1048**, 168025 (2023).

[20] C. H. Kim, S. Ahn, K. Y. Chae, J. Hooker, and G. V. Rogachev, Restoring original signals from pile-up using deep learning, Nucl. Instrum. Methods Phys. Res., Sect. A **1055**, 168492 (2023).

[21] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, A review of uncertainty quantification in deep learning: Techniques, applications and challenges, Inf. Fusion **76**, 243 (2021).

[22] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, Hands-on Bayesian neural networks–A tutorial for deep learning users, IEEE Comput. Intell. Mag. **17**, 29 (2022).

[23] K. P. Murphy, *Probabilistic Machine Learning: An Introduction* (MIT Press, Cambridge, 2022).

[24] C. H. Kim, Improving phenomenological analysis using probabilistic neural networks, Ph.D. thesis, Sungkyunkwan University, Seoul, 2024 (unpublished).

[25] A. M. Lane and R. G. Thomas, *R*-matrix theory of nuclear reactions, Rev. Mod. Phys. **30**, 257 (1958).

[26] G. M. Hale, R. E. Brown, and N. Jarmie, Pole structure of the $J^\pi = 3/2^+$ resonance in ^{5}He, Phys. Rev. Lett. **59**, 763 (1987).

[27] H. O. U. Fynbo, C. A. Diget, U. C. Bergmann, M. J. G. Borge, J. Cederkäll, P. Dendooven, L. M. Fraile, S. Franchoo, V. N. Fedosseev, B. R. Fulton, W. Huang, J. Huikari, H. B. Jeppesen, A. S. Jokinen, P. Jones, B. Jonson, U. Köster, K. Langanke, M. Meister, T. Nilsson *et al.*, Revised rates for the stellar triple-α process from measurement of ^{12}C nuclear resonances, Nature (London) **433**, 136 (2005).

[28] R. E. Azuma, E. Uberseder, E. C. Simpson, C. R. Brune, H. Costantini, R. J. de Boer, J. Görres, M. Heil, P. J. LeBlanc, C. Ugalde, and M. Wiescher, AZURE: An *R*-matrix code for nuclear astrophysics, Phys. Rev. C **81**, 045805 (2010).

[29] E. G. Adelberger, A. García, R. G. H. Robertson, K. A. Snover, A. B. Balantekin, K. Heeger, M. J. Ramsey-Musolf, D. Bemmerer, A. Junghans, C. A. Bertulani, J. W. Chen, H. Costantini, P. Prati, M. Couder, E. Uberseder, M. Wiescher, R. Cyburt, B. Davids, S. J. Freedman, M. Gai *et al.*, Solar fusion cross sections. II. The *pp* chain and CNO cycles, Rev. Mod. Phys. **83**, 195 (2011).

[30] A. Tumino, C. Spitaleri, M. La Cognata, S. Cherubini, G. L. Guardo, M. Gulino, S. Hayakawa, I. Indelicato, L. Lamia, H. Petrascu, R. G. Pizzone, S. M. R. Puglia, G. G. Rapisarda, S. Romano, M. L. Sergi, R. Spartá, and L. Trache, An increase in the ^{12}C+^{12}C fusion rate from resonances at astrophysical energies, Nature (London) **557**, 687 (2018).

[31] D. Odell, C. R. Brune, D. R. Phillips, R. J. deBoer, and S. N. Paneru, Performing Bayesian analyses with AZURE2 using BRICK: An application to the ^{7}Be System, Front. Phys. **10**, 888476 (2022).

[32] J. Bishop, C. E. Parker, G. V. Rogachev, S. Ahn, E. Koshchiy, K. Brandenburg, C. R. Brune, R. J. Charity, J. Derkin, N. Dronchi, G. Hamad, Y. Jones-Alberty, T. Kokalova, T. N. Massey, Z. Meisel, E. V. Ohstrom, S. N. Paneru, E. C. Pollacco, M. Saxena, N. Singh *et al.*, Neutron-upscattering enhancement of the triple-alpha process, Nat. Commun. **13**, 2151 (2022).

[33] H. O. Meyer, G. R. Plattner, and I. Sick, Elastic P+ ^{12}C scattering between 0.3 and 2 MeV, Z. Phys. A: Hadrons Nucl. **279**, 41 (1976).

[34] N. Otuka, E. Dupont, V. Semkova, B. Pritychenko, A. Blokhin, M. Aikawa, S. Babykina, M. Bossant, G. Chen, S. Dunaeva, R. Forrest, T. Fukahori, N. Furutachi, S. Ganesan, Z. Ge, O. Gritzay, M. Herman, S. Hlavač, K. Katō, B. Lalremruata *et al.*, Towards a more complete and accurate experimental nuclear reaction data library (EXFOR): International collaboration between nuclear reaction data centres (NRDC), Nucl. Data Sheets **120**, 272 (2014).

[35] F. Ajzenberg-Selove, Energy levels of light nuclei $A = 13$–15, Nucl. Phys. A **523**, 1 (1991).

[36] S. Artemov, E. Zaparov, and G. Nie, Asymptotic normalization factors for light nuclei from proton transfer reactions, Bull. Russ. Acad. Sci.: Phys. **67**, 1741 (2003).

[37] R. Longland, C. Iliadis, A. E. Champagne, J. R. Newton, C. Ugalde, A. Coc, and R. Fitzgerald, Charged-particle thermonuclear reaction rates: I. Monte Carlo method and statistical distributions, Nucl. Phys. A **841**, 1 (2010).

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Red Hook, NY, USA, 2017), Vol. 30.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Association for Computational Linguistics, Minneapolis, 2019), pp. 4171–4186.

[40] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, Proc. AAAI Conf. Artif. Intell. **35**, 11106 (2021).

[41] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, Transformers in vision: A survey, ACM Comput. Surv. **54**, 1 (2022).

[42] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2016), pp. 770–778.

[43] J. Lei Ba, J. R. Kiros, and G. E. Hinton, Layer normalization, arXiv:1607.06450.

[44] D. Hendrycks and K. Gimpel, Gaussian error linear units (GELUs), arXiv:1606.08415.

[45] A. Kendall and Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision? in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Red Hook, NY, USA, 2017).

[46] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, May 7–9, 2015, Conference Track Proceedings, edited by Y. Bengio and Y. LeCun (ICLR, Appleton, WI, 2015).

[47] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, A comprehensive survey on transfer learning, Proc. IEEE **109**, 43 (2021).

[48] A. Der Kiureghian and O. Ditlevsen, Aleatory or epistemic? Does it matter? Struct. Saf. **31**, 105 (2009).

[49] E. Hüllermeier and W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, Mach. Learn. **110**, 457 (2021).

[50] A. J. K. Chua and M. Vallisneri, Learning Bayesian posteriors with neural networks for gravitational-wave inference, Phys. Rev. Lett. **124**, 041102 (2020).

[51] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, On calibration of modern neural networks, in *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, New York, 2017), pp. 1321–1330.

[52] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, Can you trust your model's uncertainty? Evaluating predictive uncertainty under dataset shift, in *Advances in Neural Information Processing Systems*, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Red Hook, NY, USA, 2019).

[53] V. Kuleshov, N. Fenner, and S. Ermon, Accurate uncertainties for deep learning using calibrated regression, in *Proceedings of the 35th International Conference on Machine Learning*,

Proceedings of Machine Learning Research, Vol. 80, edited by J. Dy and A. Krause (PMLR, New York, 2018), pp. 2796–2804.

[54] Y. Gal and Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 48, edited by M. F. Balcan and K. Q. Weinberger (PMLR, New York, 2016), pp. 1050–1059.

[55] T. M. Mitchell, *Machine Learning* (McGraw-Hill, New York, 1997).

[56] D. P. Kingma and M. Welling, An introduction to variational autoencoders, Found. Trends Learn. Res. **12**, 307 (2019).

[57] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (O'Reilly Media, Sebastopol, CA, 2022).

[58] R. Andrae, Error estimation in astronomy: A guide, arXiv:1009.2755.

[59] Y. Gal, Uncertainty in deep learning, Ph.D. thesis, University of Cambridge, Cambridge, 2016.

[60] M. Valdenegro-Toro and D. S. Mori, A deeper look into aleatoric and epistemic uncertainty disentanglement, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (IEEE, Piscataway, NJ, 2022), pp. 1508–1516.

[61] B. Lakshminarayanan, A. Pritzel, and C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Red Hook, NY, USA, 2017).

[62] A. G. Wilson and P. Izmailov, Bayesian deep learning and a probabilistic perspective of generalization, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Red Hook, NY, USA, 2020), pp. 4697–4708.

[63] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, Variational inference: A review for statisticians, J. Am. Stat. Assoc. **112**, 859 (2017).