


# Neural network analytic continuation for Monte Carlo:Improvement by statistical errors

Kai-Wei Sun(孙恺伟) and Fa Wang(王垡)

**Citation:**Chin. Phys. B, 2023, 32 (**7**): 070705. DOI: 10.1088/1674-1056/accd4c
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

**What follows is a list of articles you may be interested in**

---

## ESR-PINNs: Physics-informed neural networks with expansion-shrinkage resampling selection strategies

Jianan Liu(刘佳楠), Qingzhi Hou(侯庆志), Jianguo Wei(魏建国), and Zewei Sun(孙泽玮)
Chin. Phys. B, 2023, 32 (**7**): 070702. DOI: 10.1088/1674-1056/acc1d5

## Learnable three-dimensional Gabor convolutional network with global affinity attention for hyperspectral image classification

Hai-Zhu Pan(潘海珠), Mo-Qi Liu(刘沫岐), Hai-Miao Ge(葛海淼), and Qi Yuan(袁琪)
Chin. Phys. B, 2022, 31 (**12**): 120701. DOI: 10.1088/1674-1056/ac8cd7

## A spintronic memristive circuit on the optimized RBF-MLP neural network

Yuan Ge(葛源), Jie Li(李杰), Wenwu Jiang(蒋文武), Lidan Wang(王丽丹), and Shukai Duan(段书凯)
Chin. Phys. B, 2022, 31 (**11**): 110702. DOI: 10.1088/1674-1056/ac6b1d

## Ionospheric vertical total electron content prediction model in low-latitude regions based on long short-term memory neural network

Tong-Bao Zhang(张同宝), Hui-Jian Liang(梁慧剑),Shi-Guang Wang(王时光), and Chen-Guang Ouyang(欧阳晨光)
Chin. Phys. B, 2022, 31 (**8**): 080701. DOI: 10.1088/1674-1056/ac4487

## Memristor-based multi-synaptic spiking neuron circuit for spiking neural network

Wenwu Jiang(蒋文武), Jie Li(李杰), Hongbo Liu(刘洪波), Xicong Qian(钱曦聪), Yuan Ge(葛源), Lidan Wang(王丽丹), and Shukai Duan(段书凯)
Chin. Phys. B, 2022, 31 (**4**): 040702. DOI: 10.1088/1674-1056/ac380b

---

**RAPID COMMUNICATION**

# Neural network analytic continuation for Monte Carlo: Improvement by statistical errors

Kai-Wei Sun(孙恺伟)[1] and Fa Wang(王垡)[1,2,†]

[1] *International Center for Quantum Materials, School of Physics, Peking University, Beijing* 100871, *China*
[2] *Collaborative Innovation Center of Quantum Matter, Beijing* 100871, *China*

(Received 17 February 2023; revised manuscript received 13 April 2023; accepted manuscript online 16 April 2023)

This study explores the use of neural network-based analytic continuation to extract spectra from Monte Carlo data. We apply this technique to both synthetic and Monte Carlo-generated data. The training sets for neural networks are carefully synthesized without "data leakage". We find that the training set should match the input correlation functions in terms of statistical error properties, such as noise level, noise dependence on imaginary time, and imaginary time-displaced correlations. We have developed a systematic method to synthesize such training datasets. Our improved algorithm outperforms the widely used maximum entropy method in highly noisy situations. As an example, our method successfully extracted the dynamic structure factor of the spin-1/2 Heisenberg chain from quantum Monte Carlo simulations.

**Keywords:** neural network, analytic continuation, quantum Monte Carlo

**PACS:** 07.05.Mh, 02.70.Ss **DOI:** 10.1088/1674-1056/accd4c

## 1. Introduction

Numerical analytic continuation (AC) solves the following inversion problem:

$$G(\tau) = \int \mathrm{d}\omega K(\tau,\omega)A(\omega). \tag{1}$$

The goal of AC is to extract the real frequency spectrum $A(\omega)$ from the imaginary-time correlation function $G(\tau)$, which is typically obtained by Monte Carlo simulation. The spectrum $A(\omega)$ is required to be non-negative at any $\omega$-point and subjected to certain sum rule $\int \mathrm{d}\omega A(\omega) = \text{const}$. $K(\tau,\omega)$ is the inversion kernel, the form of which varies on specific problems being handled. This study involves two kinds of inversion kernels $K(\tau,\omega)$ including $K_{\mathrm{F}}(\tau,\omega) = \mathrm{e}^{-\tau\omega}/(1+\mathrm{e}^{-\beta\omega})$ and $K_{\mathrm{S}}(\tau,\omega) = \mathrm{e}^{-\tau\omega}$. $K_{\mathrm{F}}(\tau,\omega)$ usually appears while calculating single-particle excitation spectra from measured Green's functions.[1,2] $K_{\mathrm{S}}(\tau,\omega)$ is usually involved while extracting dynamic structure factors from spin–spin correlation functions in some spin models.[3]

To carry out actual calculation, $\tau$ and $\omega$ are often discretized as $\tau_i = \tau_1,\ldots,\tau_M$, $\omega_i = \omega_1,\ldots,\omega_N$. Then the target problem can be reformulated as $G(\tau_i) = \sum_{j=1}^{N} K(\tau_i,\omega_j)A(\omega_j)\Delta\omega$. For the purpose of simplicity, $\Delta\omega$ will be absorbed to $A(\omega_j)$ by $A(\omega_j)\Delta\omega \to A(\omega_j)$ in further discussions. The sum rule is then discretized to be $\sum_{i=1}^{N} A(\omega_i)\Delta\omega = \text{const}$. It seems like a simple problem of matrix inversion at first sight but turns out to be a notoriously challenging task due to the ill-conditioned nature of this inversion problem. In almost all cases, corresponding condition numbers go far beyond the tolerance of existing computers' machine precision. Several methods are proposed to solve this problem such as the maximum entropy method (Maxent)[2] and stochastic analytic continuation (SAC).[4–6] Both of them succeed in extracting empirically correct spectra. However, these methods usually demand highly accurate simulated correlation functions $G_{\mathrm{sim}}(\tau)$.

As an emerging technique for machine learning, neural networks (NNs)[7] have experienced great success in a variety of physics-related domains. From the perspective of machine learning, analytic continuation can be viewed as a vector-to-vector prediction task, where $G(\tau_i)$ is mapped to $A(\omega_j)$. To construct a neural network capable of performing analytic continuation, both the network topology and training set should be built appropriately. The common framework for this task usually contains several steps: (1) Build a neural network. (2) Synthesize spectra $A_{\mathrm{train}}$ for training purpose. (3) Calculate $G_{\mathrm{train}}$ by the forward mapping $A \to G$. Noting that the forward mapping is well-conditioned, thus $G_{\mathrm{train}}$ can be exactly determined. (4) Train the network using the dataset pair $(G_{\mathrm{train}}, A_{\mathrm{train}})$ so that spectra predicted from $G_{\mathrm{train}}$ closely match $A_{\mathrm{train}}$. (5) When developing and testing NNs, synthesize testing set $(G_{\mathrm{test}}, A_{\mathrm{test}})$ and evaluate the performance of the trained NN on it. When using neural networks based analytic continuation (NNAC) in actual tasks, apply the trained network to predict spectra $A_{\mathrm{pred}}$ from simulated correlation functions $G_{\mathrm{sim}}$ generated by Monte Carlo simulations. To mimic real-world simulated data, noises are usually added to corre-

[†]Corresponding author. E-mail: wangfa@pku.edu.cn

© 2023 Chinese Physical Society and IOP Publishing Ltd 

lation functions obtained from synthetic spectra such as $G_{\text{train}}$ and $G_{\text{test}}$.

In a relatively early study, Hongkee Yoon[8] and co-authors designed a network mainly based on fully-connected-layers (FCLs).[7] In their research, both training and testing sets are obtained from synthetic Gaussian-type multi-peak spectra. Noises of Gaussian distribution are added to $G_{\text{train}}$ and $G_{\text{test}}$. The trained NN performs well in the testing set as the predicted spectra are very close to synthetic testing spectra. Several different network structures[9–12] trained on similar Gaussian-type datasets are also proposed. In addition to synthetic datasets, NNAC is also examined on some exactly solvable models such as one-dimensional transverse-field Ising model[13] and harmonic oscillator linearly coupled to an ideal environment.[14] In these two studies, artificial training sets ($G_{\text{train}}$, $A_{\text{train}}$) are generated from exactly solved correlation functions and corresponding spectra. Different spectra in the training set correspond to different parameter values in the Hamiltonian being studied. Target spectra $A_{\text{pred}}$ are predicted from simulated correlation function $G_{\text{sim}}$ using Monte Carlo techniques. Reference [13] points out that the neural network's prediction performance can be improved by adding uniform noises to the exactly solved Green's functions at each imaginary time in the training set.

Theoretically, we have no knowledge about precise forms of spectra to be predicted before target spectra are actually predicted. That is because the knowledge of Gaussian-type spectra is not expected to be known before prediction. This is actually an intriguing topic dubbed "data leakage"[15] in the field of machine learning. Data leakage occurs when information is used in the training process but not expected to be available at the prediction time. All aforementioned articles about NNAC have the issue of data leakage at some levels. In practice, we usually apply numerical analytical continuation to models that are not exactly solvable, where it is not possible to construct training sets by exactly solved spectra.

To design the training set, hints from experiments or traditional AC approaches such as Maxent should also be explored. It should be mentioned that NNAC is useful even when spectra are already obtained from Maxent: NNAC performs better at least in highly-noisy cases as described in Ref. [14]. This topic will also be elaborated upon in this paper. In general, domain knowledge,[16,17] especially possible spectrum peak shapes, should be incorporated when designing the training set as much as feasible but without data leakage. We then expect the trained NN to generalize[18,19] well enough to handle unobserved correlation functions like $G_{\text{test}}$ and $G_{\text{sim}}$.

Intuitively, people expect better prediction of spectra by incorporating more information. Monte Carlo simulations can provide more information beyond the measured correlation functions, such as the statistical errors of $G(\tau)$. Specifically, they can provide information regarding two aspects of statistical errors: the measured errors $R(\tau_i)$ of $G(\tau_i)$ at each $\tau_i$, and the covariance of correlation functions at different imaginary times.

This work avoids data leakage while synthesizing the training sets and incorporates information of statistical errors to improve the performance of NNAC. With these means, NNAC has the potential to be a usable algorithm in practical applications and a significant component in the Monte Carlo-analytic continuation tool chain. In Section 2, NNAC of kernel $K_{\text{F}}(\tau,\omega)$ is examined on synthetic data, where datasets synthesized from spectra of different types of shapes are addressed. In Section 3, NNAC of kernel $K_{\text{S}}(\tau,\omega)$ is applied to a one-dimensional Heisenberg chain as a real-world example of an AC problem. Conclusions are presented in the final section. We demonstrate that NNAC can acquire knowledge embedded within the inversion kernel, as detailed in Appendix A, and discuss the performance of NNAC when the input correlation functions are accurate in Appendix B. In Appendix C, we show how to incorporate information from Maxent into NNAC to enhance its robustness.

## 2. NNAC on synthetic datasets

In this section, we design and test NNs on synthetic datasets. Principles for generating training sets will be developed. We first discuss three types of datasets, the training framework, as well as the actual training process. Noise level matching between the training and the testing set is then explored. The resulting spectra are then compared with those from Maxent. The impact of measured noise shapes and time-displaced correlation is then investigated.

### 2.1. Preparation of dataset

Multi-peak spectra $A(\omega)$ are produced by summing over single peaks $F(\omega)$.

$$A(\omega)=\frac{1}{Z}\sum_i F_i(\omega). \tag{2}$$

In the formula above, $Z$ is a scaling constant ensuring that $A(\omega)$ obeys the sum rule. In this section, we assume $\int \mathrm{d}\omega A(\omega)=1$ for convenience. This paper involves three distinct peak types: asymmetric exponential power (ASEP), skew Gaussian (skew), and Lorentz. The ASEP single-peak curve reads

$$F^{\text{ASEP}}(\omega)=\begin{cases} h\exp\left[-\left(\dfrac{m-\omega}{a_1}\right)^{b_1}\right], & \omega<m; \\ h\exp\left[-\left(\dfrac{\omega-m}{a_2}\right)^{b_2}\right], & \omega\geq m. \end{cases} \tag{3}$$

In the above formula, $h$, $m$, $a_1$, $a_2$, $b_1$, $b_2$ are all control parameters. In this study, we set $m \in [-5,5]$, $a_1, a_2 \in [0.3,3]$, $b_1, b_2 \in [1,3]$, $h \in [0.2,1]$. The skew peak takes the form

$$F^{\text{Skew}}(\omega) = \begin{cases} 0, & z \le 0; \\ \frac{h}{az} \exp\left(-\frac{y^2}{2}\right), & z > 0. \end{cases} \tag{4}$$

$z(\omega) = 1 - k\frac{\omega - m}{a}$ and $y = \frac{1}{k}\ln(z)$. Control parameters are $m \in [-2,2]$, $a \in [0.5,1]$, $k \in [-1,1]$, and $h \in [0.2,1]$. The Lorentz curve takes the relatively simple form

$$F^{\text{Lorentz}}(\omega) = h\frac{1}{(\omega^2 - a^2)^2 - g^2\omega^2}, \tag{5}$$

where $g \in [1,2]$, $a \in [2,4]$ and $h \in [0.2,1]$. In this study, we investigate spectra containing one to four peaks. At least $10^5$ samples are generated for each peak number by randomly selecting control parameters. In other words, one single dataset includes at least $4 \times 10^5$ samples. Training and testing sets of the same peak type are independently created.

The ASEP-type dataset has the most control parameters among these three types and thus contains a greater diversity of spectra while not explicitly containing spectra of skew-type or Lorentz-type dataset. We expect the neural network to learn from the ASEP-type dataset and generalize effectively to achieve good performance on the other two datasets. It should be noted that, unlike in some previous studies, we will not examine Gaussian-type spectra here, as they are explicitly included in ASEP-type dataset when $b_1 = b_2 = 2$ and $a_1 = a_2$. This explicit inclusion case does not frequently occur in real-world AC tasks and the performance of NNAC will be overestimated in the case of Gaussian-type testing sets.

The imaginary time $\tau \in [0,16]$ is discretized uniformly into 512 pieces and the frequency domain $\omega \in [-15,15]$ is discretized into 1024 pieces. $\beta$ is fixed to be 16 in the Fermion kernel $e^{-\tau\omega}/(1+e^{-\beta\omega})$.

### 2.2. Training framework

Convolution neural networks (CNNs)[20] will be employed in this work. FCL-based neural networks are also evaluated in the early stage of this study, which proves inferior to CNNs. Involvement of residual modules[21] or deep layer aggregation[22] also does not prove to make significant improvements. In the case of deep layer aggregation, both iterative deep aggregation and hierarchical deep aggregation are attempted. Based on the aforementioned factors, we employ the neural network shown in Fig. 1. At first the 512-length $G(\tau_i)$ is transferred to a $p$-length vector via an FCL (labeled "Dense") and then reshaped to be a $1 \times p$ matrix. This matrix can be regarded as a specific image that can be naturally processed by convolution layers. Next, this image is passed to a $q$-channel one-dimensional convolution layer "Conv1d", followed by the activation layer "Swish". Within the "Conv1d" layer, convolution kernels of size $1 \times 3$ are used. Within the activation layer, the activation function named "Swish"[23] is used. This activation function is both non-monotonic and smooth and may improve the overall performance of the neural network compared to the commonly used ReLU[24] activation function according to Ref. [23]. This "convolution → activation" process will be carried out $n$ times. The $q$-channel image is then compressed by an average-pooling layer[25] and flattened to be a $pq/2$-long vector. The flattened vector will be mapped to a 1024-long vector by another "Dense" layer. Ultimately, the "SoftMax" layer will present predictions of the spectra where the sum rule $\sum_j A(\omega_j) = 1$ is naturally satisfied after this softmax operation. Tricks to reduce overfitting such as dropout[26] are not adopted here. Instead, we recommend enlarging the training set when signs of overfitting emerge since it is rather cheap to acquire data from synthetic spectra.

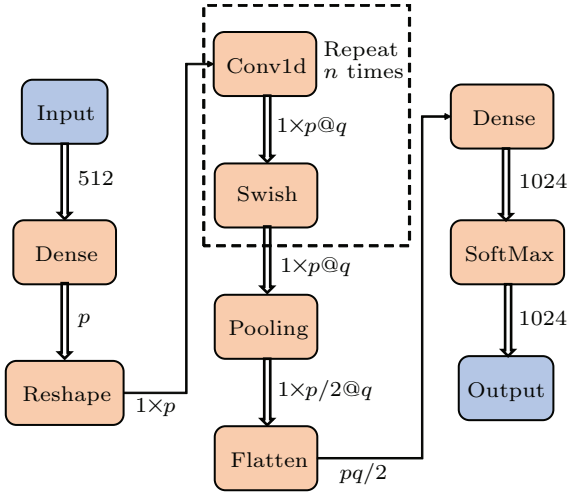



**Fig. 1.** The convolution-based structure of the neural network used in this work. Hyper-parameters are chosen to be $n = 8$, $p = 64$ and $q = 64$ in actual training process.

Hyper-parameters are chosen to be $n = 8$, $p = 64$, and $q = 64$. To select appropriate hyper-parameters, we build an additional ASEP-type validation set, on which to evaluate NN trained by the ASEP-type training set. When selecting hyper-parameters, the trade-off between performance and training time is taken into consideration.

We use Kullback–Leibler divergence (KLD)[27] as the loss function, which takes the form

$$D_{\text{KL}}(A_{\text{true}}||A_{\text{pred}}) = -\sum_j A_{\text{true}}(\omega_j)\ln\frac{A_{\text{true}}(\omega_j)}{A_{\text{pred}}(\omega_j)}. \tag{6}$$

KLD measures the difference (more precisely, relative entropy) between the true distribution $A_{\text{true}}$ and the predicted distribution $A_{\text{pred}}$, which makes it a natural choice in this task. Other commonly-used loss functions include mean absolute

error (MAE) and mean squared error (MSE) as shown below:

$$\mathrm{MAE}(A_{\mathrm{true}},A_{\mathrm{pred}}) = \frac{1}{N}\sum_{j=1}^{N}\left|A_{\mathrm{true}}(\omega_j) - A_{\mathrm{pred}}(\omega_j)\right|, \quad (7)$$

$$\mathrm{MSE}(A_{\mathrm{true}},A_{\mathrm{pred}}) = \frac{1}{N}\sum_{j=1}^{N}\left[A_{\mathrm{true}}(\omega_j) - A_{\mathrm{pred}}(\omega_j)\right]^2. \quad (8)$$

KLD also has the property of positivity as MAE and MSE.

Empirically, spectra from NNs with MSE loss are often smoother than those from NNs with MAE loss since MSE punish large spectrum difference more severely. In this study, we didn't observe discernible difference in the performance between MSE-loss and KLD-loss NNs.

NNs are programmed using Keras toolkits[28] with Tensorflow[29] backends. The Adam[30] optimizer is used for gradient descent. The early-stopping trick is utilized during training. The training process terminates when KLD measured on the validation set does not drop for 20 epochs, where the validation set is generated in the same manner as the training set. Trained weights are then restored to the epoch with the lowest KLD. Each training task will be repeated at least 5 times with different random seeds. KLDs shown in this paper are averaged among NNs trained with different seeds.

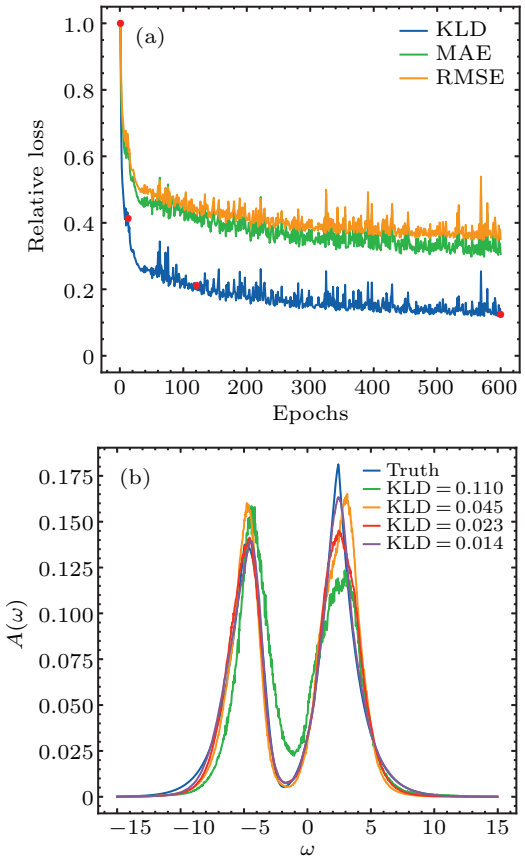



**Fig. 2.** Tracking the training process. (a) Relative losses, including KLD, MAE, and RMSE, vs. the number of trained epochs. This so-called relative loss is defined as "loss after this epoch" divided by "loss after the first epoch". (b) A typical example of the convergence process of one predicted spectrum to the true spectrum as KLD decreases. Selected checkpoints are labeled by red dots in (a).

The training process is depicted in Fig. 2, where both the training set and the testing set are of ASEP-type. Errors at noise level $10^{-3}$ are introduced to $G_{\mathrm{train}}$ and $G_{\mathrm{test}}$ (the concept of noise level will be discussed later). Relative values of three statistics measured on the testing set are tracked throughout the training process in Fig. 2(a). We track RMSE $= \sqrt{\mathrm{MSE}}$ instead of MSE itself because RMSE shares the same dimension as MAE and KLD. Relative loss in this figure is defined as "loss after this epoch" divided by "loss after the first epoch". In Fig. 2(b) we show an example from the testing set of how one predicted spectrum becomes closer to the true spectrum at different KLD levels. Selected checkpoints are indicated by red dots in Fig. 2(a). While visualizing the training process, we only use 1000 samples for each epoch because statistics converge too quickly for visualization if the entire training set containing $4\times10^5$ samples is used. The complete training set will be used in actual AC tasks hereafter.

In this study, model training on an RTX3060 graphics card takes approximately 20 min on average. This is acceptable in the majority of circumstances, especially in contrast to the amount of time saved in the Monte Carlo simulation if highly accurate correlation functions are not incorporated.

### 2.3. Noise level matching

Correlation functions measured from Monte Carlo simulation inevitably contain statistical errors. To mimic simulated errors, Gaussian-type noises are added to $G(\tau_i)$ by $G(\tau_i) \to G(\tau_i) + R(\tau_i)$, where $R(\tau_i) \sim N(0,\sigma^2)$. Four different noise levels are investigated in this work, $\sigma = 10^{-4}$, $10^{-3}$, $3\times10^{-3}$, $10^{-2}$. $\sigma$ in this formula can also be interpreted as the absolute average of noises. At this stage, we assume $G(\tau_i)$ to be independently measured for each $i$. In real-world NNAC-based tasks, noises of $G_{\mathrm{sim}}$ are measured from Monte Carlo simulation, and noises of the training set should be carefully arranged accordingly. Besides, the noise level of the testing set should be the same as the simulated data to mimic real-world tasks.

To design the training set, a natural question arises as to how we should set noise level $\sigma_{\mathrm{train}}$ of the training set when the noise level $\sigma_{\mathrm{test}}$ of the testing set is known. We train NNs by training sets of different $\sigma_{\mathrm{train}}$ and apply these NNs on testing sets of different $\sigma_{\mathrm{test}}$. Corresponding results are shown in Table 1 and Fig. 3. Table 1 contains KLDs of spectra predicted from testing sets with different noise levels $\sigma_{\mathrm{test}}$ by NNs trained by training sets with different $\sigma_{\mathrm{train}}$. The smallest KLD in each line (marked red) is obtained when noise levels of the training set and the testing set match ($\sigma_{\mathrm{train}} = \sigma_{\mathrm{test}}$). Performance degrades but remains acceptable when $\sigma_{\mathrm{train}}$ increases and $\sigma_{\mathrm{train}} > \sigma_{\mathrm{test}}$ while the opposite is not true when $\sigma_{\mathrm{train}} < \sigma_{\mathrm{test}}$. For instance, KLD is relatively small when $(\sigma_{\mathrm{train}},\sigma_{\mathrm{test}}) = (10^{-2},10^{-4})$ but is large and unsatisfactory when $(\sigma_{\mathrm{train}},\sigma_{\mathrm{test}}) = (10^{-4},10^{-2})$. That's because

information of ASEP($\sigma = 10^{-4}$) is somehow "contained" in ASEP($\sigma = 10^{-2}$): for each curve in ASEP($\sigma = 10^{-4}$) we may find similar samples with similar noises in ASEP($\sigma = 10^{-2}$) if datasets are large enough given noises are randomly selected, whereas the converse is not true. We train NNs with different noise levels and use them to predict one sample of $G(\tau_i)$ from the testing set with $\sigma_{\text{test}} = 3 \times 10^{-3}$ and $\sigma_{\text{test}} = 10^{-2}$, which are presented in Figs. 3(a) and 3(b), respectively. The resulted spectra become closer to the ground truth when $\sigma_{\text{train}}$ is closer to $\sigma_{\text{test}}$. In Fig. 3(b), incorrect and unstable peaks are predicted by the NNs trained with $\sigma_{\text{train}} = 10^{-4}$ or $10^{-3}$, whose KLDs are large correspondingly as seen in Table 1.

**Table 1.** KLDs of spectra predicted from testing sets with different $\sigma_{\text{test}}$ by NNs trained by training sets with different $\sigma_{\text{train}}$. In each line, the smallest KLD (marked red) is obtained when $\sigma_{\text{train}} = \sigma_{\text{test}}$. To determine the errors of the KLDs in the table, we train NNs with at least 10 distinct random seeds and calculate the statistical uncertainty of KLDs of spectra predicted by these NNs.

| $\sigma_{\text{test}}$ \ $\sigma_{\text{train}}$ | $10^{-4}$ | $10^{-3}$ | $3 \times 10^{-3}$ | $10^{-2}$ |
|---|---|---|---|---|
| $10^{-4}$ | 0.0137(3) | 0.0151(4) | 0.0181(2) | 0.0280(1) |
| $10^{-3}$ | 0.0172(1) | 0.0164(4) | 0.0185(2) | 0.0280(1) |
| $3 \times 10^{3}$ | 0.045(2) | 0.0268(3) | 0.0217(1) | 0.02854(9) |
| $10^{-2}$ | 0.31(2) | 0.148(6) | 0.060(1) | 0.0350(1) |

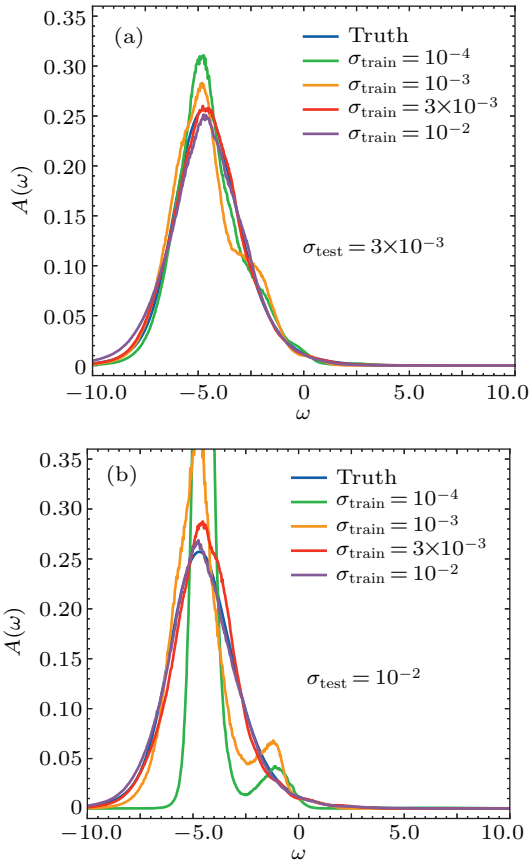



**Fig. 3.** Illustration of noise level matching. Ground truths in both subfigures are the same curve. (a) Prediction of spectra from testing set with $\sigma = 3 \times 10^{-3}$ by NNs trained with different $\sigma_{\text{train}}$. The best spectrum is obtained when $\sigma_{\text{train}} = \sigma_{\text{test}} = 3 \times 10^{-3}$. (b) Prediction of spectra from testing set with $\sigma = 10^{-2}$ by NNs trained with different $\sigma_{\text{train}}$. The best spectrum is obtained when $\sigma_{\text{train}} = \sigma_{\text{test}} = 10^{-2}$. The predicted spectrum contains unstable peaks at wrong locations when $\sigma_{\text{train}} = 10^{-4}$ or $3 \times 10^{-3}$.

Note that in this part, data leakage is not intentionally avoided: the training set and the testing set are both of ASEP type. With the same $\sigma_{\text{test}}$, KLD differences caused by different $\sigma_{\text{train}}$ may be relatively small and taking datasets with different line shapes may introduce unnecessary complexity, resulting in unsolid or even incorrect conclusions. From another perspective, we expect NNs to use the knowledge learned from the training set to predict correct spectra in actual tasks. The performance will be usually slightly weakened if the line shapes of the testing set and training set are different. Therefore, we expect the NNs of proper $\sigma_{\text{train}}$ to at least achieve good results on the testing set with the same line shape. The KLD results here do not represent the actual performances of the NNs in practical tasks.

### 2.4. Comparison with Maxent

With the knowledge of noise level matching, $G_{\text{train}}$ will be designed to have the same noise level as $G_{\text{test}}$ in this work hereafter and we are now ready to compare NNAC with traditional AC methods like Maxent. We train NNs with ASEP training sets and use them to predict ASEP-type, skew-type and Lorentz-type spectra, respectively. Corresponding outcomes are depicted in Fig. 4. Figures 4(a), 4(b) and 4(c) show KLDs of spectra predicted by these two methods on ASEP, skew, and Lorentz dataset respectively. Error bars of KLDs are omitted in this and subsequent figures to make graphs more comprehensible as they are relatively small. Typical predicted results at noise level $3 \times 10^{-3}$ are shown in Figs. 4(d), 4(e) and 4(f) of three peak types. The performance of NNAC is comparable with Maxent at the lowest noise level $10^{-4}$ but outperforms Maxent significantly at relatively high noise levels. The improvement of the prediction effect is also obvious when the training set and testing set are not of the same spectrum type. In spectrum examples depicted in Figs. 4(d), 4(e) and 4(f), peak locations are precisely predicted by NNAC but Maxent didn't provide accurate peak locations at this noise level. In some frequencies, Maxent may even give incorrect signals of peaks. Peak heights predicted by NNAC are also more accurate and closer to ground truths than Maxent's.

Spectra from Maxent in this section about kernel $K_{\text{F}}(\tau, \omega)$ are calculated mainly based on the software "TRIQS/maxent"[31] so that results can be easily checked. Various $\alpha$-choosing algorithms are evaluated, where $\alpha$ is the penalty coefficient of the entropy term in the Maxent objective function.[2] Among these algorithms discussed in Ref. [31], "$\chi_2$-curvature", which is analogous to $\Omega$-Maxent,[32] and "Bryan" algorithms greatly outperform others in terms of KLD in tasks of interest. Between these two, "$\chi_2$-curvature" is marginally superior to the Bryan algorithm. In this way, we use "$\chi_2$-curvature" in this work to ensure a level playing field for Maxent.

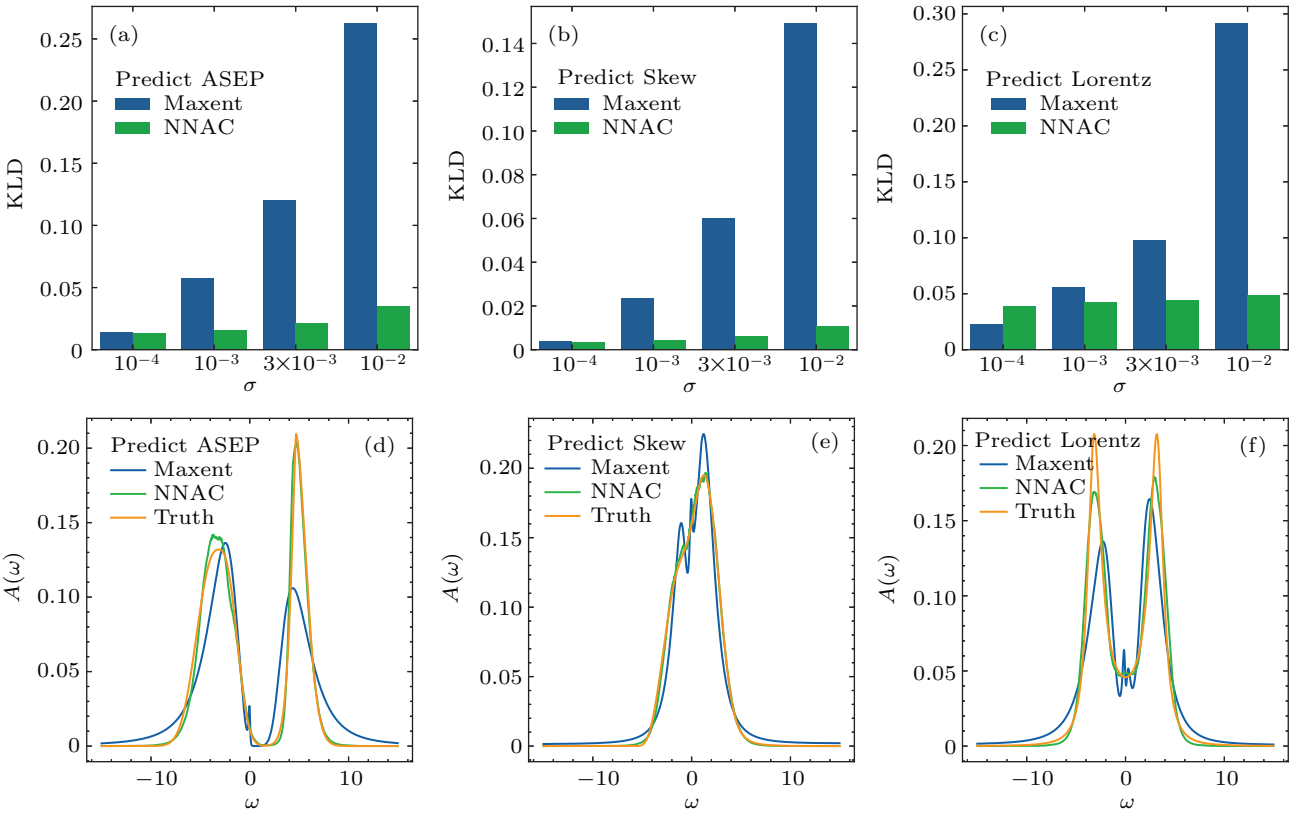



**Fig. 4.** Comparison with Maxnet. NNs are trained by the ASEP dataset and applied on three different testing sets: ASEP, skew, and Lorentz. (a)–(c) KLD predicted results of ASEP, skew, Lorentz datasets respectively at different noise levels. (d)–(f) Typical predicted spectra at the noise level $3\times10^{-3}$ by Maxent and NNAC. The ground truth is also shown for comparison. The performance of NNAC is comparable with Maxent when the dataset contains low-level noise but outperforms Maxent at high-level noise even if NNs are not trained by the dataset of the same type as the testing set.

### 2.5. Influence of noise dependency on imaginary time

In the preceding discussion, noise $R(\tau_i)$ at each $\tau_i$ is assumed to be sampled from the same Gaussian distribution and has the same variance, which is rarely the case in Monte Carlo simulation. We introduce the noise-shape-multiplier $\lambda(\tau)$ to investigate the influence of noise dependency on imaginary time and assume $R(\tau_i)\sim N(0,\sigma(\tau_i)^2)$, $\sigma(\tau_i)=\lambda(\tau_i)\sigma$. We refer to this dependency as "noise shape" hereafter. These multipliers satisfy $\frac{1}{\beta}\int_0^\beta \lambda(\tau)\mathrm{d}\tau=1$ to ensure that datasets with the same $\sigma$ but different noise shapes are at approximately the same noise level. $\lambda(\tau)$ of four distinct linear shapes labeled A, B, C, and D are displayed in Fig. 5(a).

To demonstrate the impact of noise shape and how to appropriately arrange noises in the training set, we train NNs by ASEP-type training sets with equal noise ($\lambda(\tau)=1$) and noise shape A, respectively. These trained NNs are implemented on skew-type testing sets with noise shape A. Corresponding measured KLDs are presented in Fig. 5(b). Spectra examples at noise level $3\times10^{-3}$ are shown in in Fig. 5(c).

Origins of different performances by different noise shapes can be, to some extent, explained by permutation feature importance (PFI),[33] despite the fact that neural networks are typically seen as black boxes. To calculate PFI, we rearrange $G(\tau_i)$ randomly over samples on one certain time piece $\tau_i$ in the testing set and PFI at this time piece is defined by how much the resulted KLD increases. PFI difference between NNs trained by datasets of linear noise shapes and equal-noise dataset is defined as $[\mathrm{PFI}^T(\tau_i)-\mathrm{PFI}^\mathrm{E}(\tau_i)]/[\mathrm{PFI}^T(\tau_i)+\mathrm{PFI}^\mathrm{E}(\tau_i)]$. $\mathrm{PFI}^\mathrm{E}(\tau_i)$ denotes PFI from NNs trained by equal-noise dataset and $\mathrm{PFI}^T(\tau_i)$ denotes PFI from NNs trained by dataset of some other noise shape, where $T\in[\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}]$. Resulted relative PFI differences are shown in Fig. 5(d). Moving average of adjacent five points is carried out to make curves smoother and clearer. Relative PFI curves and $\lambda(\tau)$ curves increase or decrease in the opposite direction, which means NNs assign large feature importance on imaginary time pieces where $G(\tau_i)$ are less noisy.

It should be emphasized that measured correlation functions do not often have linear-type noise shapes. Instead, they are frequently of exponential-like shapes. However, things can become more subtle in the case of exponential noise shape, when it becomes more difficult to disentangle the effects of different noise levels and noise shapes. In light of these concerns, we only examine linear-type noise shapes here, and it is believed that physical images are similar in other scenarios.

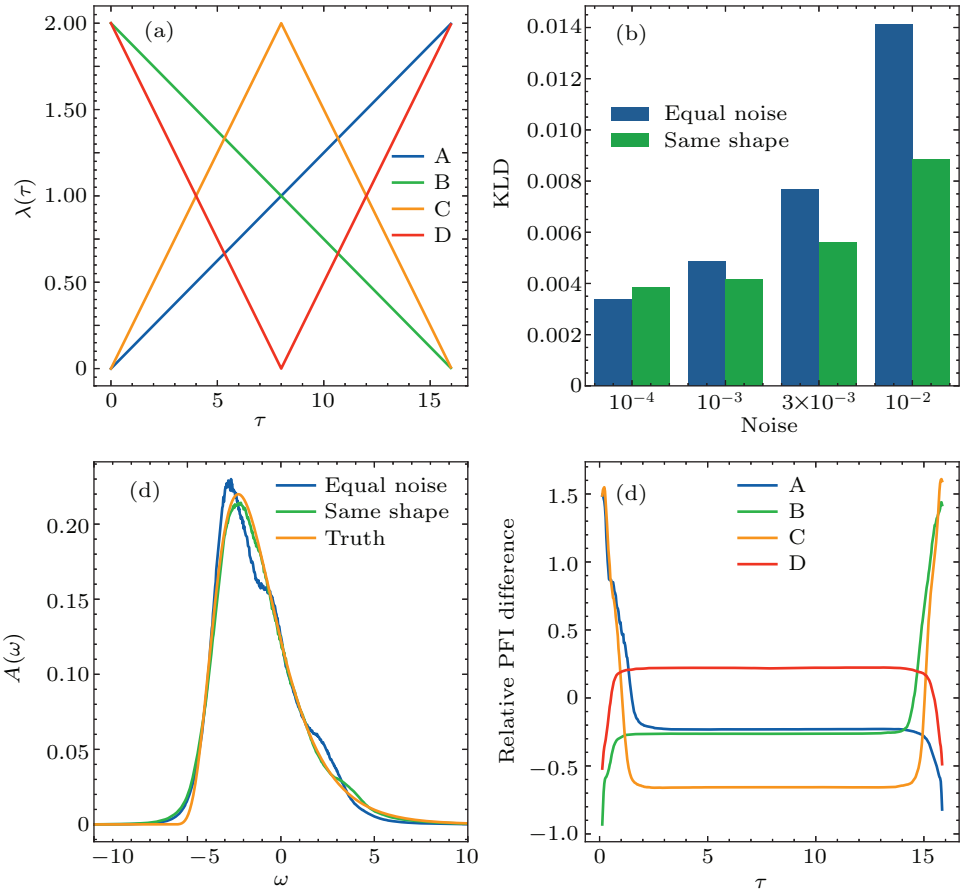



**Fig. 5.** Influence of linear noise shapes. (a) Four types of shape multiplier $\lambda(\tau)$. (b) Noises of the testing set are of shape A. Two neural networks are trained by the training set of equal noise ($\lambda(\tau) = 1$) and noise shape A, respectively. KLDs of trained neural networks are compared on the shape-A testing set at different noise levels. (c) Typical spectra predicted from $G(\tau)$ of shape-A noises ($\sigma = 3 \times 10^{-3}$) by neural networks trained by equal-noise and shape-A training sets, respectively. (d) The relative difference in PFI between the neural network trained on a training set with linearly-shaped noise and the neural network trained on a training set with uniformly-shaped ($\lambda(\tau) = 1$) noise at various imaginary times.

### 2.6. Influence of time-displaced correlation

In the previous discussion, we assumed that the correlation functions $G(\tau_i)$ on different imaginary times $\tau_i$ are independently measured, which means there is no mutual correlation between correlation functions at different imaginary times. In actual Monte Carlo simulations, independent measurements can indeed be achieved, but they come with significant computational overhead. Generally speaking, the correlation functions at different imaginary times are usually measured simultaneously on a single Markov chain, and thus there may be strong mutual correlations between them. The impact of mutual correlations can be measured by the covariance matrix $\Sigma$, defined as

$$\Sigma_{ij} = \langle G(\tau_i)G(\tau_j)\rangle - \langle G(\tau_i)\rangle\langle G(\tau_j)\rangle. \tag{9}$$

Here $\langle\cdots\rangle$ denotes the average over Monte Carlo samples. In practical analytic continuation tasks, the covariance matrix $\Sigma$ can be obtained before synthesizing the training set. If it is required that spectra in the training set or the testing set also satisfy a certain covariance matrix $\Sigma$, errors should be obtained from the joint normal distribution, i.e., $R \sim N(0, \Sigma)$:

1) Perform singular value decomposition (SVD) on the covariance matrix $\Sigma = UDV^{\mathrm{T}}$. Since the covariance matrix is real-symmetric, we have $U = V$.

2) Define the diagonal matrix $d$ by $d_{ii} = \sqrt{D_{ii}}$.

3) Define another matrix $L = Ud$.

4) Generate the vector $r$, where each element is a sample from the standard normal distribution, $r_i \sim N(0, 1)$.

5) Calculate $R = Lr$.

Of course, one can also use the Cholesky decomposition $\Sigma = LL^{\mathrm{T}}$ to directly generate the matrix $L$, but it is widely accepted that the numerical stability of SVD is better than that of Cholesky decomposition. Maxent requires to perform SVD on the covariance matrix to obtain the inverse of the singular values. Given that some singular values may be close to zero, this operation can exhibit numerical instability. Conversely, NNAC circumvents this possible numerical instability arising from the inversion of singular values.

To illustrate the influences of time-displaced correlation, we create the toy covariance matrix for the testing set by

$$\Sigma_{ij} = \frac{\sigma^2}{1 + |i-j|^{1/\gamma}}. \tag{10}$$

In this work, we will investigate correlation matrices with condition numbers being $10^3$, $10^6$, $10^9$, and $10^{12}$ respec-

tively by adjusting $\gamma$. $U(\tau)$ are generated at four noise levels $\sigma \in [10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}]$.

NNs are trained by ASEP-type datasets and are to be applied to skew-type testing sets with various noise levels and condition numbers. Training sets are designed in two manners: they may have zero correlation ($R \sim N(0, \Sigma_0)$) or the same correlation as the testing set($R \sim N(0, \Sigma)$). $\Sigma_0$ have the same diagonal elements as $\Sigma$ but all off-diagonal elements of $\Sigma_0$ are 0. In Fig. 6(a), condition number of the testing set is fixed to be $10^{12}$. NNs are trained by the dataset with or without time-displaced correlation on each noise level. As illustrated, the influence of $\tau$-correlation is not significant at low noise levels but correlation mismatching may lead to incorrect prediction at high noise levels. In Fig. 6(b), the noise level of the testing set (and the training set, as well) is fixed to be $10^{-2}$, where KLDs are lower when the condition number is larger. The reason may be that $R(\tau_i)$ are dominated by only a few singular values of $\Sigma$, whose pattern of noises is relatively easy to be learned by NNs. Spectrum examples are shown in Fig. 6(c) with noise level $10^{-2}$ and condition number $10^{12}$, which contain predicted spectra by NNs trained with zero or the same correlation as the testing set, as well as the ground truth. Clearly, the predicted spectra contain wrong peaks at the wrong locations when the time-displaced correlation is not matched.

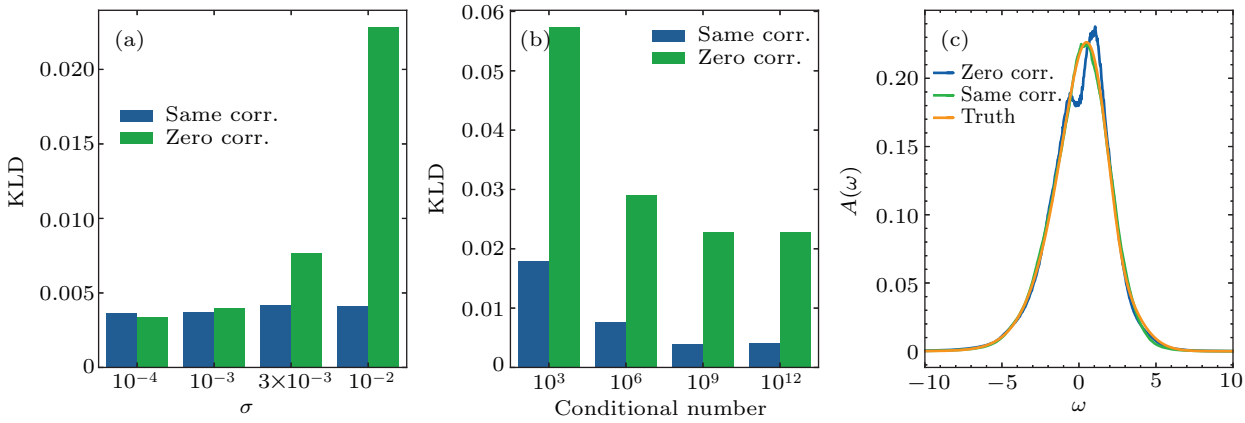



**Fig. 6.** Illustration of influences of time-displaced correlation of $G(\tau_i)$. Noise levels of the training set and the testing set are matched. (a) The condition number of the correlation matrix is fixed to be $10^{12}$. NNs trained by datasets without correlation may give wrong predictions if $G(\tau_i)$ in the testing set is correlated, especially when the noise level is high. (b) Noise levels are fixed to be $10^{-2}$. KLDs are shown with respect to different condition numbers. (c) Spectra examples with condition number $10^{12}$ and noise level $10^{-2}$.

## 3. NNAC on Heisenberg chain

In this section, NNAC is carried out to extract dynamic structure factors of the spin-1/2 anti-ferromagnetic Heisenberg chain of length $L$, which reads

$$H = J\sum_{i=1}^{L} \boldsymbol{S}_i \cdot \boldsymbol{S}_{i+1}, \tag{11}$$

where $\boldsymbol{S}_i$ represents a spin located on site $i$. Periodic boundary condition is assumed, i.e., $\boldsymbol{S}_{L+1} = \boldsymbol{S}_1$. Imaginary-time-displaced spin–spin correlation of $z$-component is measured by stochastic series expansion[34]

$$G_{i,j}(\tau) = \langle \mathrm{e}^{\tau H} S_i^z \mathrm{e}^{-\tau H} S_j^z \rangle, \tag{12}$$

$$G_k(\tau) = \frac{1}{L}\sum_{i,j} G_{i,j}(\tau)\, \mathrm{e}^{-\mathrm{i}(r_i - r_j)k/L}. \tag{13}$$

$G_{i,j}(\tau)$ is time-displaced spin–spin correlation of $z$-component between spin $i$ and spin $j$. Target correlation function $G_k(\tau)$ in the wave-vector domain is then calculated via Fourier transformation. $r_i$ denotes the location of spin $i$, where the lattice constant is set to be 1. $J$ is used as the energy unit. We set the inverse temperature $\beta = 1$ in the Monte Carlo simulation. In this work we will focus on $k = \pi$ and $G_k(\tau)$ will be represented by $G(\tau)$ for the sake of simplicity. Then the AC task reads $G(\tau) = \int \mathrm{d}\omega \mathrm{e}^{-\tau\omega} A(\omega)$, where $A(\omega)$ is the target dynamic structure factor. The corresponding sum rule is obtained by setting $\tau = 0$, i.e., $\int \mathrm{d}\omega A(\omega) = G(0)$.

The same NN structure and hyper-parameters are used as in the previous section. Frequency $\omega$ takes the range $\omega \in [-10, 10]$. The time domain and the frequency domain are discretized into 512 and 1024 pieces respectively as before. The spectrum of the Heisenberg chain can be regarded as a sum of $\delta$ functions at zero temperature. These $\delta$ functions broaden as temperature increases. We perform quantum Monte Carlo simulation on a 32-site Heisenberg chain, where $\delta$ functions are dense enough on the required energy scale $\Delta\omega \sim 0.02$ so that a smooth spectrum can be obtained. The stochastic series expansion approach with loop-update[34] algorithm is used in the simulation. Spin–spin correlation is measured every 100 update steps so that auto-correlation can be ignored. The covariance matrix $\Sigma$ is measured by $\Sigma_{ij} = [\langle G(\tau_i)G(\tau_j)\rangle - \langle G(\tau_i)\rangle\langle G(\tau_j)\rangle]/(N_\mathrm{s} - 1)$, where $N_\mathrm{s}$ is the number of independent samples.

Spin–spin correlation functions are measured using dif-

ferent numbers of Monte Carlo samples to create datasets of different noise levels. In this section, noise levels are represented by relative statistical errors of $G(0)$, which takes range from $3.8\times10^{-3}$ to $3.6\times10^{-2}$. Simulated $G(\tau)$ are divided by corresponding $G(0)$ before being fed into neural networks so that the sum rule is restored to $\int \mathrm{d}\omega A(\omega)=1$. Then the "SoftMax" layer results in the correct sum rule and the scale of extracted spectra will be recovered accordingly by multiplying with $G(0)$.

First, we will consider the case where there exist mutual correlations between simulated correlation functions at different imaginary times. In each Monte Carlo measurement step, we simultaneously measure spin–spin correlation functions at each imaginary time. Condition numbers of measured covariance matrices range from about $10^{17}$ to $10^{18}$. As described in Subsection 2.6, we incorporate the joint normal distribution $R\sim N(0,\Sigma_0)$ or $R\sim N(0,\Sigma)$ to generate errors $R$ of the training set. The diagonal elements of $\Sigma_0$ are the same as those of $\Sigma$, while all off-diagonal elements are 0.

In Fig. 7, we show predicted spectra at different noise levels. The predicted spectra are labeled "Zero Corr." when mutual correlations between imaginary times are not incorporated in the training set ($R\sim N(0,\Sigma_0)$). When mutual correlations are considered while generating the training set, we label predicted spectra as "Same Corr.". The black dashed line in the figure corresponds to the spectrum given by Okamoto *et al.*[35] They used the finite-temperature Lanczos method (FTLM) to calculate dynamic structure factors of an antiferromagnetic Heisenberg chain of length 24, where some artificial peak broadening was added to results. Correlation functions obtained from independent Monte Carlo simulations may be slightly different, so we repeated Monte Carlo simulations with the same update and measurement steps at least 5 times and performed analytic continuation accordingly to estimate the variance of the predicted spectra. The blue solid line in each subfigure of Fig. 7 represents the averaged spectrum, while the light blue area corresponds to the range within one standard deviation for each imaginary time.

As shown in Figs. 7(a) and 7(d), the difference between cases "Zero Corr." and "Same Corr." is more significant when the noise level is lower. The predicted spectrum is inaccurate and has a larger variance when not considering imaginary-time mutual correlations. This difference decreases as the noise decreases, as shown in Figs. 7(c) and 7(f). When the noises of the input correlation function are relatively small, there is no significant difference between the predicted spectra in these two cases.

It is worth noting that although the curve calculated by FTLM shown in Fig. 7 is obtained from a Heisenberg chain with $L=24$ rather than $L=32$ considered in this work, at relatively high accuracies, results of NNAC are in good agreement with the FTLM spectrum, at least in terms of peak position and peak width within one standard deviation.

Due to the high condition number of the covariance matrix $\Sigma$, Maxent cannot generate reasonable spectra in this situation. To compare the performance of Maxent and NNAC, we independently measure correlations at each imaginary time, so the off-diagonal elements of the corresponding

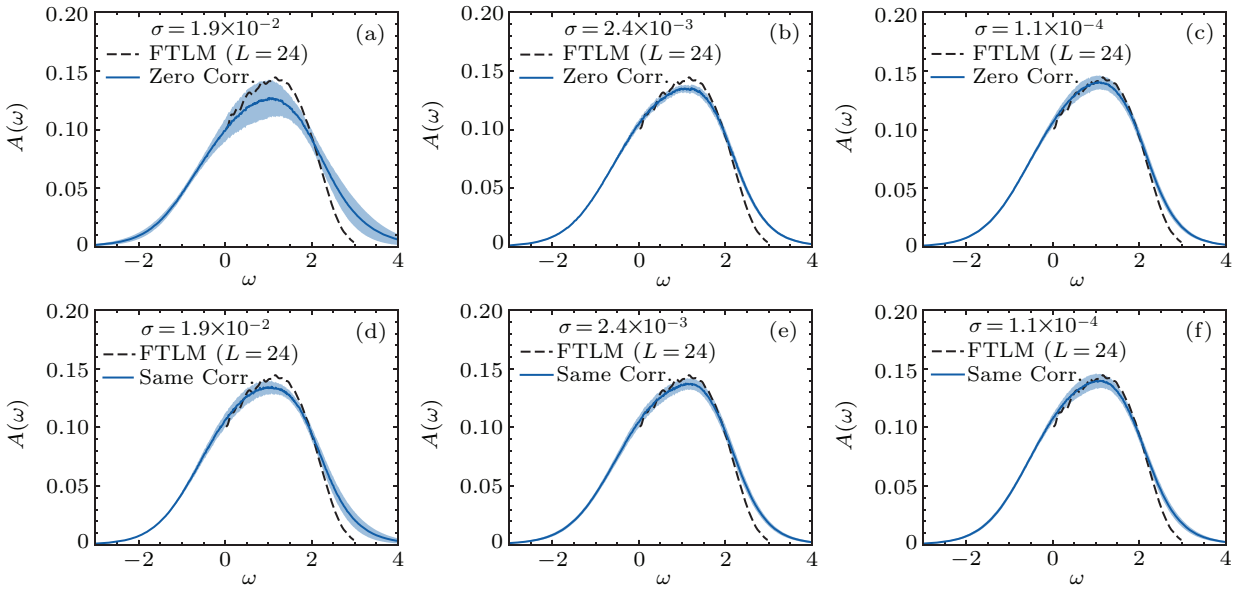



**Fig. 7.** Dynamic structure factors of the one-dimensional antiferromagnetic Heisenberg chain predicted by NNAC. Monte Carlo simulations with the same update steps and measurement steps are performed independently at least 5 times to obtain different spin–spin correlation functions, then NNAC is performed accordingly. The blue solid line in each subfigure represents the averaged spectrum predicted by NNAC from several independently measured correlation functions. The light blue area indicates the range of one standard deviation of the spectrum. Subfigures (a), (b), and (c) correspond to the training process without considering imaginary time mutual correlations, while subfigures (d), (e), and (f) correspond to the training process that considers imaginary time mutual correlations. Data of the black dashed line comes from Okamoto *et al.*[35] They use the FTLM method to solve the dynamic structure factor of a one-dimensional Heisenberg chain of length 24.

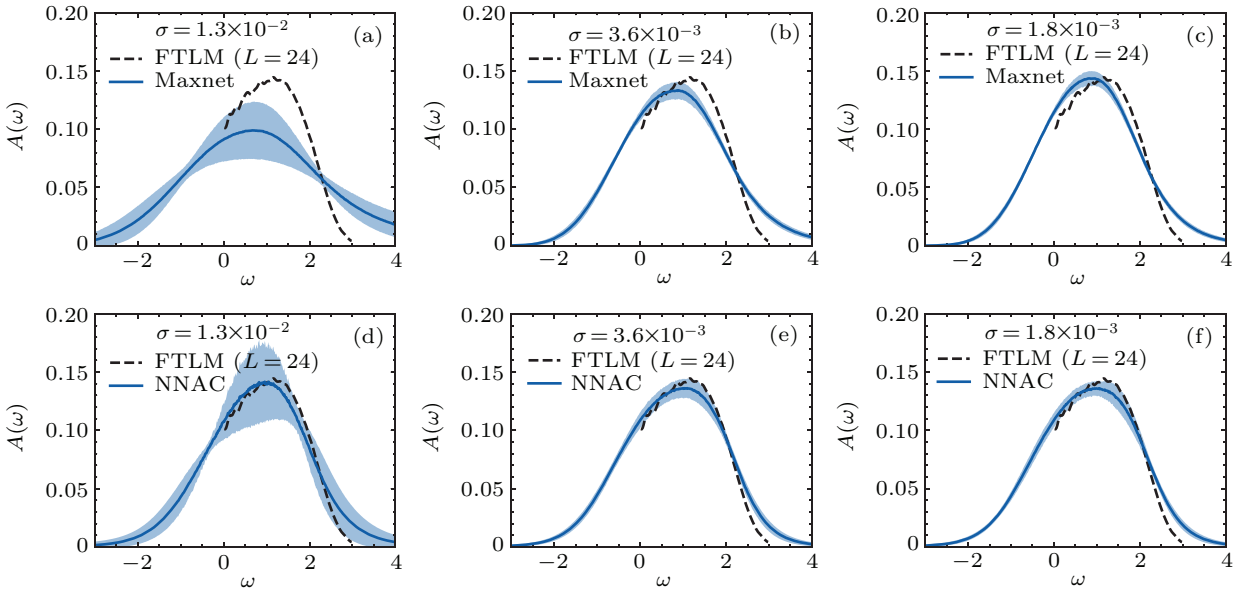



**Fig. 8.** Spectra predicted by Maxent and NNAC in the absence of imaginary time mutual correlation. Monte Carlo simulations with the same update steps and measurement steps are performed independently at least 5 times to obtain different spin–spin correlation functions, then NNAC is performed accordingly. The blue solid line in each subfigure represents the averaged spectrum predicted by NNAC from several independently measured correlation functions. The light blue area indicates the range of one standard deviation of the spectrum. Subfigures (a), (b), and (c) correspond to the training process without considering imaginary time mutual correlations, while subfigures (d), (e), and (f) correspond to the training process that considers imaginary time mutual correlations. At the noise levels involved in this figure, the NNAC results are better than those of Maxent. Data of the black dashed line comes from Okamoto *et al.*[35] They use the FTLM method to solve the dynamic structure factor of a one-dimensional Heisenberg chain of length 24.

covariance matrix are 0. The analytic continuation task we need to deal with returns to the situation described in Subsection 2.5: the correlation function has no imaginary time mutual correlation, but noises have imaginary time dependence. As before, we perform Maxent and NNAC at different noise levels and show relevant results in Fig. 8. At the different noise levels involved in the figure, spectra predicted by NNAC are more accurate than those given by Maxent, which agrees with Subsection 2.4.

## 4. Conclusions

Applications of neural network-based analytic continuation were discussed in this paper. Numerical experiments are carried out on both synthetic datasets and Monte Carlo data. The main conclusion is that a NN can learn from a carefully designed training set and make good predictions on spectra without data leakage, which surpasses Maxent in highly noisy cases. To ensure that the neural network acquires adequate knowledge to predict the target spectral functions, the training dataset should comprise a sufficient number of diverse spectral functions. Incorporating information of measured statistical errors leads to better prediction of spectra. $G(\tau_i)$ of the training set should match those of simulated correlation functions in terms of noises at each $\tau_i$ and time-displaced correlation.

While acceptable, the time required for NNAC is relatively long compared to Maxent. Improving the efficiency of model training may be a fruitful area for future investigation. It may be possible to apply the idea of transfer-learning[36] here so that we do not need to train a model from scratch for each target spectrum but rather begin with a pre-trained model. A more valuable and ambitious goal is to train a model that is general to any spectrum. The input to this model should probably be the simulated correlation functions and the accompanying covariance matrices, which contain most (if not all) information needed to perform analytic continuation.

## Appendix A: Inversion kernel learned by NNAC

In Subsection 2.5, we studied how the noise shape affects predictions of NNAC by analyzing PFI. Here we also use PFI to study feature importance for the spectrum at a specific frequency. The feature importance of the correlation function at imaginary time $\tau_i$ for the spectrum frequency $\omega_j$ is denoted as $I(\tau_i, \omega_j)$ and can be calculated in following steps:

1) Train the NN by the training set $(G_{\text{train}}, A_{\text{train}})$.

2) Predict spectra $A_{\text{pred}}$ from testing correlation functions $G_{\text{test}}$.

3) Traverse each imaginary time $\tau_i$:

3-a) Permute features at $\tau_i$ of the testing correlation functions $G_{\text{test}}$. Use trained NN to predict spectra $A_{\text{perm}}$ from permuted $G_{\text{test}}$.

3-b) For each sample in the testing set, calculate

$$\Delta A^s(\tau_i, \omega_j) = |A^s_{\text{perm}}(\omega_j) - A^s_{\text{test}}(\omega_j)| - |A^s_{\text{pred}}(\omega_j) - A^s_{\text{test}}(\omega_j)|, \tag{A1}$$

where $A^s$ is the $s^{\text{th}}$ sample in the testing spectra.

3-c) Take average over samples

$$I(\tau_i,\omega_j)=\frac{1}{N_s}\sum_s \Delta A^s(\tau_i,\omega_j), \tag{A2}$$

where $N_s$ is the number of samples in the testing set.

3-d) Rescale $I(\tau_i,\omega_j)$

$$I(\tau_i,\omega_j)\leftarrow\frac{I(\tau_i,\omega_j)}{\frac{1}{512}\sum_{i=1}^{512} I(\tau_i,\omega_j)}. \tag{A3}$$

This step ensures that sums of feature importance at each frequency are the same.

4) Rescale feature importances into the range $[0,1]$,

$$I(\tau_i,\omega_j)\leftarrow\frac{I(\tau_i,\omega_j)}{\max_{i,j} I(\tau_i,\omega_j)}. \tag{A4}$$

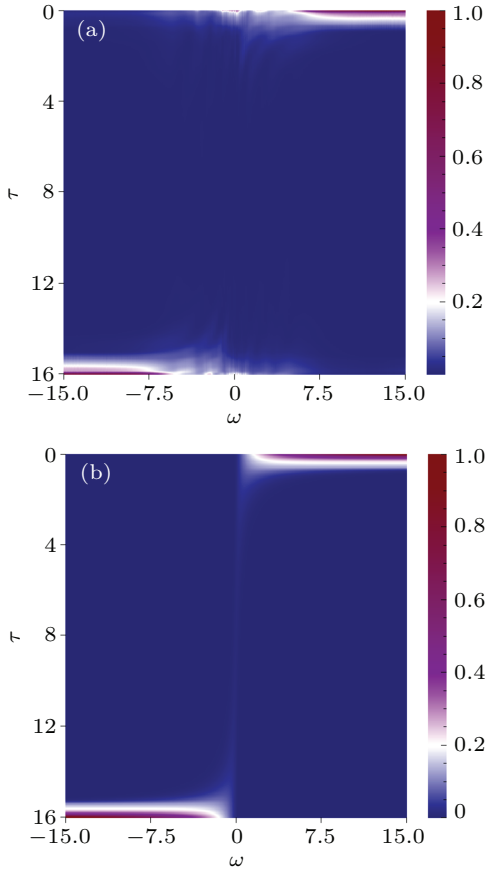



**Fig. A1.** (a) Feature importance for each specific frequency. The training set is of ASEP-type, and the testing set is of skew-type. The noise level of correlation functions is $10^{-4}$. Here we don't consider the mutual correlation between imaginary times. The inversion kernel is $K_F(\tau,\omega)$. (b) The inversion kernel rescaled by Eqs. (A3) and (A4). To denoise the data, we have performed moving average with a window length of 20 points in the imaginary time direction in both subfigures.

Here we use ASEP as the training set, skew as the testing set, and $K_F(\tau,\omega)$ as the inversion kernel. Noises of input correlation functions are set to be $10^{-4}$ and we don't incorporate mutual correlation between imaginary times. Resulted feature importance $I(\tau_i,\omega_j)$ is shown in the left part of Fig. A1. The inversion kernel $K_F(\tau,\omega)$ rescaled by Eqs. (A3) and (A4) is shown in the right part of Fig. A1. Indeed, the trained NN has successfully learned some knowledge contained in the inversion kernel: when $\omega$ is small, the trained NN relies on $G(\tau)$ with large $\tau$; when $\omega$ is large, the trained NN relies on $G(\tau)$ with small $\tau$.

## Appendix B: When noise levels are low

Some algorithms like DMRG or tensor network can provide accurate correlation functions at the precision of a double-precision floating-point number, which is approximately $10^{-16}$. To handle the analytic continuation from accurate correlation functions, Maxent usually requires the manual addition of noises to correlation functions. As an extension of this work, we consider the performance of NNAC when the error level is extremely low.

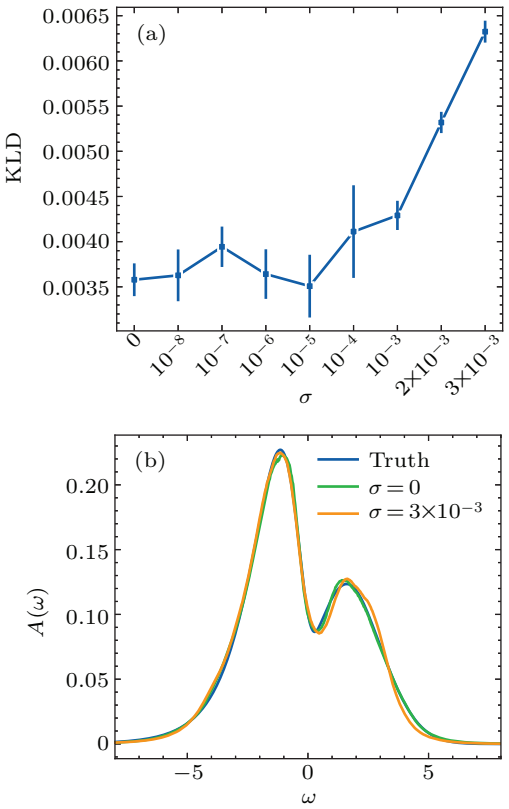



**Fig. B1.** Performance of NNAC when noise levels are very low. The training set is of ASEP-type. The testing set is of skew-type. (a) KLDs of spectra predicted by NNAC at different noise levels. (b) One typical example of predicted spectra.

As in Subsection 2.3, we train the NN by the ASEP-type training set and evaluate the trained NN on the skew-type testing set at different noise levels. The mutual correlations between imaginary times are not incorporated. Corresponding results are shown in Fig. B1. We show KLDs of spectra predicted by NNAC at different noise levels in Fig. B1(a). One typical example is presented in Fig. B1(b). On one hand, when the noise level is very low, NNAC can still be effective (extra noises are not needed); on the other hand, when the noise level is less than $10^{-5}$, reducing the noise does not bring a significant improvement to NNAC's performance. When dealing

with high input accuracy, the NN's performance may be limited by the size of the network architecture and the training set.

## Appendix C: Robust prediction with the help of Maxent

As mentioned earlier, NNAC requires the training set to be sufficiently abundant and diverse. A potential risk of NNAC is that there is not enough overlap between spectra in the training set and the target spectrum. In this section, we attempt to incorporate knowledge from Maxent into the training set to obtain more robust predictions, ensuring that the performance of NNAC is at least not worse than Maxent. The basic idea is to generate an extra set of spectra from the spectrum generated by Maxent and incorporate this extra dataset into the training set.

Suppose the correlation function obtained from simulation is $G_{\rm sim}(\tau)$, and the spectrum produced by Maxent is $A_{\rm maxent}(\omega)$. As the first step, we need to generate a set of spectra $A_{\rm deform}$ based on the Maxent spectrum $A_{\rm maxent}(\omega)$:

1) Decompose the spectrum $A_{\rm maxent}(\omega)$ into individual peaks to obtain a series of single-peak curves $F_i(\omega)$, where $i = 1, 2, \ldots, M$.

2) Repeat $N_{\rm deform}$ times to obtain a sufficient number of samples:

2-a) Deform each single-peak curve $F_i(\omega)$ to obtain $J_i(\omega)$.

2-b) Sum up deformed curves $J_i(\omega)$ to get $Q(\omega) = \frac{1}{Z}\sum_i J_i(\omega)$, where $Z$ is a constant to ensure that $Q(\omega)$ satisfies the corresponding sum rule.

2-c) Add $Q(\omega)$ to the set of spectra $A_{\rm deform}$.

The above process involves two important operations: one is peak separation, and the other is deformation of single-peak curves. To carry out peak separation, we use the differential evolution algorithm to fit the Maxent-generated spectrum $A_{\rm maxent}(\omega)$ with a multi-peak ASEP-type curve. Fitting residuals are assigned to each single-peak curve to ensure the sum of single-peak functions recovers the original spectrum $A_{\rm maxent}(\omega)$. The steps of the peak separation are as follows:

1) Denote the trial spectrum as $A_{\rm trial}(\omega) = \frac{1}{Z}\sum_{i=1}^{M} F_i^{\rm ASEP}(\omega)$, where $F^{\rm ASEP}(\omega)$ is defined in Eq. (3). The single-peak curve $F^{\rm ASEP}(\omega)$ has 6 control parameters, so the trial spectrum has $6M$ control parameters.

2) Minimize MSE$(A_{\rm maxent}(\omega), A_{\rm trial}(\omega))$ to obtain optimized spectrum $A_{\rm optimal}(\omega)$.

3) Denote fitting residuals as $A_{\rm residual}(\omega) = A_{\rm maxent}(\omega) - A_{\rm optimal}(\omega)$. Assign residuals to each single-peak curve by

$$F_i^{\rm residual}(\omega) = \frac{(\omega-\omega_i^{\rm c})^6}{\sum_{j=1}^{M}(\omega-\omega_j^{\rm c})^6} A_{\rm residual}(\omega), \tag{C1}$$

where $\omega_i^{\rm c}$ is the peak location of $F_i^{\rm ASEP}(\omega)$. Separated single peak curves are given by $F_i^{\rm maxent}(\omega) = F_i^{\rm ASEP}(\omega) + F_i^{\rm residual}(\omega)$, where $i = 1, 2, \ldots, M$.

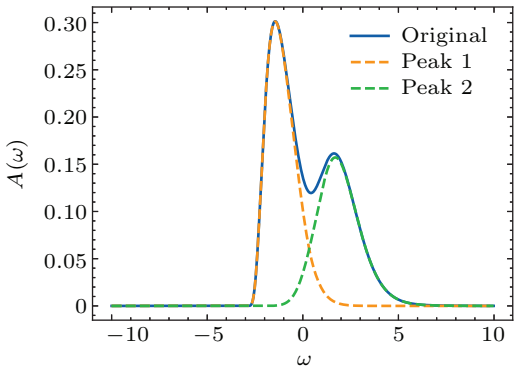



**Fig. C1.** An example of peak separation. The spectrum to be separated is the blue solid line labeled "Original", which is of skew-type. The yellow and green dashed lines represent the two single-peak curves after separation. The sum of these two single-peak curves reproduces the original double-peak spectrum.

In Fig. C1, we show an example of peak separation with $M = 2$. The blue solid line represents a skew-type spectrum to be separated, and the yellow and green dashed lines represent the two single-peak curves after separation, labeled as peak 1 and peak 2, respectively.

To deform the single-peak curve, we define four operations, namely "raise", "move", "shrink" and "distort" as shown in Fig. C2. Blue solid lines represent single-peak curves before deformation and yellow dashed lines represent single-peak curves after deformation. In fact, the peak to be deformed in Fig. C2 is "peak 2" after peak separation in Fig. C1.

The operation "raise" is defined as $F_{\rm after}(\omega) = r_{\rm raise} F_{\rm before}(\omega)$, where $r_{\rm raise}$ is the cofficient. This operation aims to adjust the peak height. In Fig. C2(a), we set $r_{\rm raise} = 1.2$. The operation "move" is defined as $F_{\rm after}(\omega) = F_{\rm before}(\omega - \omega_{\rm move})$. This operation adjusts the peak location without changing the shape. In Fig. C2(b), we set $\omega_{\rm move} = 1$. The operation "shrink" is defined as $F_{\rm after}(\omega) = F_{\rm before}(\omega^{\rm c} + r_{\rm shrink}(\omega - \omega^{\rm c}))$, where $\omega^{\rm c}$ is the peak location of the curve $F_{\rm before}(\omega)$ and $r_{\rm shrink}$ is the shrinking coefficient. This operation can be performed separately on the left half ($\omega < \omega^{\rm c}$) and the right half($\omega > \omega^{\rm c}$). In Fig. C2(c), we set the shrinking coefficient of the left to be $r_{\rm shrink}^{\rm left} = 0.5$ and don't deform the right half. The shrinking operation aims to change the peak width, and if the left and right shrinking coefficients are different, it also changes the skewness of the single-peak curve. Given the distortion coefficient $r_{\rm deform}$, the distortion operation shown in Fig. C2(d) is performed as follows:

1) Rescale the curve by $\tilde{F}_{\rm before}(\omega) = F_{\rm before}(\omega)/F_{\rm max}$, where $F_{\rm max} = \max_\omega F_{\rm before}(\omega)$.

2) Define $\alpha = \ln(r_{\rm deform})/\ln(0.5)$ and deform $\tilde{F}_{\rm before}(\omega)$ by

$$\tilde{F}_{\rm after}(\omega) = \begin{cases} 0.5[2\tilde{F}_{\rm before}(\omega)]^{\alpha}, & \tilde{F}_{\rm before}(\omega) < 0.5, \\ 1 - 0.5[2 - 2\tilde{F}_{\rm before}(\omega)]^{\alpha}, & \text{otherwise.} \end{cases} \tag{C2}$$

As in the "shrink" case, different distortion coefficients can also be assigned to the left half and the right half of the single-peak curve. In Fig. C2(d), the distortion coefficient for the left half is $r^{\text{left}}_{\text{distort}} = 0.2$ and the right half is not distorted ($r^{\text{right}}_{\text{distort}} = 0.5$). If distortion coefficients for the left half and the right half are the same, this operation changes the kurtosis of the targeted single-peak curve.

We will show how to obtain a more robust training set by incorporating the Maxent-generated spectrum for the AC task of a skew-type double-peak spectrum. First, we define three datasets: LASEP (left ASEP), RASEP (right ASEP), and DMS (deformed maxent spectra). Similar to Eqs. (2) and (3), spectra in LASEP and RASEP are still based on single-peak ASEP-type curves generated from random control parameters. The parameter selecting ranges for LASEP are $m \in [-3,0]$, $a_1, a_2 \in [0.3,5]$, $b_1, b_2 \in [1,3]$, and $h \in [0.2,1]$. For RASEP the ranges are $m \in [0,3]$, $a_1, a_2 \in [0.3,5]$, $b_1, b_2 \in [1,3]$, and $h \in [0.2,1]$. Note that peak locations $\omega^{\text{c}}$ of spectra in LASEP are less than 0, and peak locations of spectra in RASEP are greater than 0. Spectra in DMS is the set of spectra $A_{\text{deform}}$ deformed from the Maxent-generated spectrum, where deformation coefficients are randomly selected in ranges $r_{\text{raise}} \in [0.3,1]$, $\omega_{\text{move}} = [-0.5,0.5]$, $r^{\text{left}}_{\text{shrink}}, r^{\text{right}}_{\text{shrink}} \in [0.25,1.5]$, and $r^{\text{left}}_{\text{distort}}, r^{\text{right}}_{\text{distort}} \in [0.2,0.8]$. Predictions of NNs trained by various training sets are shown in Fig. C3.

As before, we add noises to the true correlation function $G_{\text{true}}(\tau)$ to mimic simulated correlation functions $G_{\text{sim}}(\tau) = G_{\text{true}}(\tau) + R(\tau)$, $R(\tau) \sim N(0,\sigma^2)$. We generate at least five different "simulated" correlation functions for each noise level with different random seeds to evaluate the standard error of resulted KLDs of predicted spectra. The structure and training process of NNs are the same as in Section 2 but the inverse temperature is adjusted to be $\beta = 4$ to avoid unnecessary complexities, where Maxent can generate reasonable spectra.

Figures C3(a), C3(d), and C3(g) compare the performance of NN trained by DMS with that of Maxent. Figure C3(a) shows KLDs of spectra predicted by these two methods at different noise levels. Even though spectra in DMS are generated based on Maxent-generated spectra, the NN trained with DMS can still provide more accurate predictions than Maxent itself. We present examples of the predicted spectra at two noise levels in Figs. C3(d) and C3(g).

Figures C3(b), C(e), and C3(h) compare the performance of NNs trained by LASEP and LASEP+DMS. LASEP doesn't contain peaks with peak locations greater than 0 ($\omega^{\text{c}} > 0$), and the ground truth contains one peak that $\omega^{\text{c}} > 0$. Apparently, the NN can not learn enough knowledge to provide good predictions. However, if DMS is added to the training set, which means incorporating the information from Maxent, the NN can make accurate predictions, at least not worse than Maxent itself. Similarly, Figures C3(c), C3(f), and C3(i) compare the results of NNs trained by DMS+RASEP and RASEP itself. When the training set only contains RASEP, spectra predicted by the neural network are inaccurate with large KLDs. When DMS is added to the training set, the performance is at least not worse than Maxent.

If a training set does not contain enough information for NNs to make good predictions, DMS can be added to ensure that NNAC is at least not worse than Maxent. In other words, DMS makes NNAC more safe and more robust.

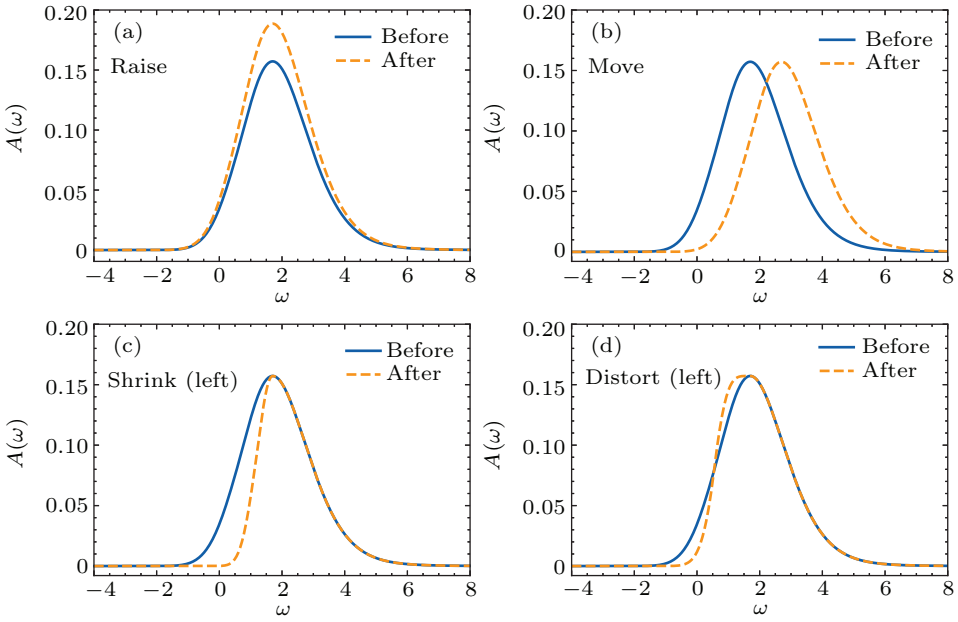



**Fig. C2.** Four types of deformation operations: raise, move, shrink, and distort. (a) Raise: this operation change peak height. (b) Move: this operation changes peak location. (c) Shrink: this operation changes the peak width. Different shrinking coefficients can be defined for the left half and the right half, respectively. (d) Distort: this operation changes the kurtosis of the single-peak curve. Different distortion coefficients can be defined for the left half and the right half, in which case the skewness also changes.

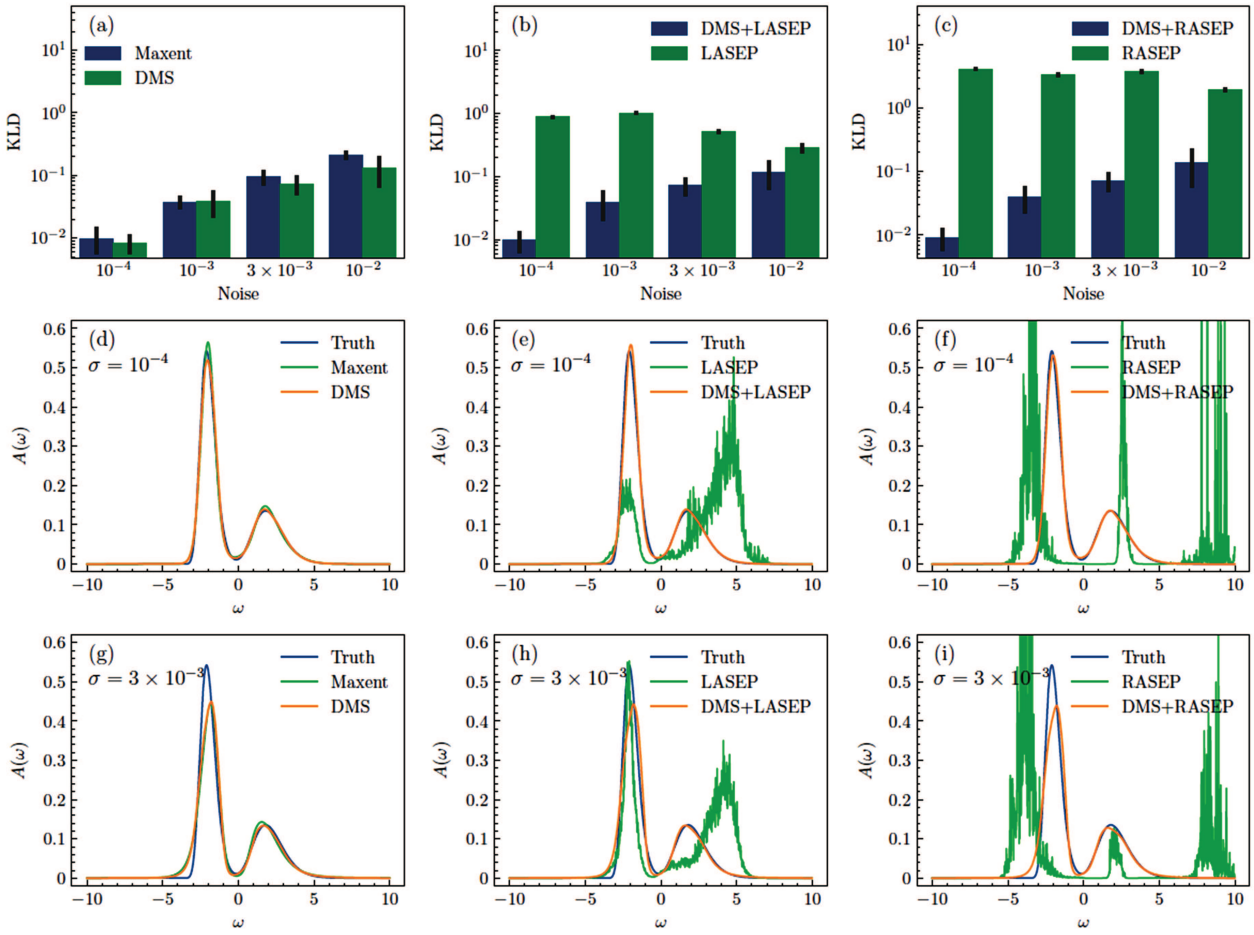



**Fig. C3.** This figure compares the predictions from neural networks trained on 5 different training sets and Maxent. These five training sets are DMS (distorted maxent spectra), LASEP (left ASEP), RASEP (right ASEP), DMS+LASEP, and DMS+RASEP. The spectra in DMS are generated by deforming the Maxent-generated spectrum. The spectra in LASEP are of ASEP-type, but peak locations of all single peaks are less than 0. The spectra in the RASEP dataset are also of ASEP-type, but the peak locations of all single peaks are greater than 0. (a) KLDs of spectra predicted by NN trained by DMS and Maxent. The DMS-trained NN outperforms Maxent at all four noise levels involved. (d) and (g) Examples of the predicted spectra by these two methods at two noise levels. (b) KLDs of spectra predicted by NNs trained by DMS+LASEP and LASEP. (e) and (h) Examples of the predicted spectra from these two methods at two noise levels. (c) KLDs of spectra predicted by NNs trained by DMS+RASEP and RASEP. (f) and (i) Examples of the predicted spectra from these two methods at two noise levels.

## Acknowledgements

FW acknowledges support from the National Natural Science Foundation of China (Grant Nos. 12274004 and 11888101). Quantum Monte Carlo simulations are performed on TianHe-1A of National Supercomputer Center in Tianjin.

## References

[1] White S, Scalapino D, Sugar R and Bickers N 1989 *Phys. Rev. Lett.* **63** 1523
[2] Silver R N, Sivia D S and Gubernatis J E 1990 *Phys. Rev. B* **41** 2380
[3] Henelius P, Sandvik A W, Timm C and Girvin S 2000 *Phys. Rev. B* **61** 364
[4] Sandvik A W 1998 *Phys. Rev. B* **57** 10287
[5] Shao H and Sandvik A W 2023 *Phys. Rep.* **1003** 1
[6] Sandvik A W 2016 *Phys. Rev. E* **94** 063308
[7] LeCun Y, Bengio Y and Hinton G 2015 *Nature* **521** 436
[8] Yoon H, Sim J H and Han M J 2018 *Phys. Rev. B* **98** 245101
[9] Arsenault L F, Neuberg R, Hannah L A and Millis A J 2017 *Inverse Problems* **33** 115007
[10] Xie X, Bao F, Maier T and Webster C 2021 *Analytic continuation of noisy data using adams bashforth residual neural network* Tech. rep. Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States)
[11] Huang D and Yang Y F 2022 *Phys. Rev. B* **105** 075112
[12] Zhang R, Merkel M E, Beck S and Ederer C 2022 *Phys. Rev. Research* **4** 043082
[13] Yao J, Wang C, Yao Z and Zhai H 2022 *Machine Learning: Science and Technology* **3** 025010
[14] Fournier R, Wang L, Yazyev O V and Wu Q 2020 *Phys. Rev. Lett.* **124** 056401
[15] Kaufman S, Rosset S, Perlich C and Stitelman O 2012 *ACM Transactions on Knowledge Discovery from Data (TKDD)* **6** 1
[16] Jordan M I and Mitchell T M 2015 *Science* **349** 255
[17] Domingos P 2012 *Communications of the ACM* **55** 78
[18] Giles C L and Maxwell T 1987 *Appl. Opt.* **26** 4972
[19] Novak R, Bahri Y, Abolafia D A, Pennington J and Sohl-Dickstein J 2018 arXiv preprint arXiv:1802.08760
[20] Albawi S, Mohammed T A and Al-Zawi S 2017 *International conference on engineering and technology (ICET)* (IEEE) pp. 1–6
[21] He K, Zhang X, Ren S and Sun J 2016 *Deep residual learning for image recognition Proceedings of the IEEE conference on computer vision and pattern recognition* pp. 770–778
[22] Yu F,Wang D, Shelhamer E and Darrell T 2018 *Deep layer aggregation Proceedings of the IEEE conference on computer vision and pattern recognition* p. 2403–2412
[23] Ramachandran P, Zoph B and Le Q V 2017 arXiv preprint arXiv:1710.05941
[24] Agarap A F 2018 arXiv preprint arXiv:1803.08375

[25] Iosifidis A and Tefas A 2022 *Deep learning for robot perception and cognition*
[26] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 *The journal of machine learning research* **15** 1929
[27] Joyce J M 2011 *Kullback-leibler divergence International encyclopedia of statistical science* (Springer) pp. 720–722
[28] Chollet F, *et al.* 2015 Keras https://keras.io
[29] Abadi M, *et al.* 2015 *Tensor Flow: Large-scale machine learning on heterogeneous systems software*
[30] Kingma D P and Ba J 2014 arXiv preprint arXiv:1412.6980
[31] Kraberger G J, Triebl R, Zingl M and Aichhorn M 2017 *Phys. Rev. B* **96** 155128
[32] Bergeron D and Tremblay A M S 2016 *Phys. Rev. E* **94** 023303
[33] Altmann A, Toloşi L, Sander O and Lengauer T 2010 *Bioinformatics* **26** 1340
[34] Sandvik A W 1999 *Phys. Rev. B* **59** R14157
[35] Okamoto S, Alvarez G, Dagotto E and Tohyama T 2018 *Phys. Rev. E* **97** 043308
[36] Pan S J and Yang Q 2010 *IEEE Transactions on knowledge and data engineering* **22** 1345